AMENDMENTS TO THE DRAWINGS

Applicant has amended FIGs. 6-10 to include the legend "Prior Art." No new matter has been added.

The attached replacement sheets designated as FIGs. 6-10 replace the original sheets designated as FIGs. 6-10.

REMARKS

In the Official Action mailed on **23 August 2005** the Examiner reviewed claims 67-70. The abstract was objected to for being too long. The drawings were objected to. Claims 67 and 69 were objected to under 35 U.S.C. §101 because they are directed to non-statutory subject matter. Claims 69-70 were rejected under 35 U.S.C. §103(a) as being unpatentable over Gligor (USPub 2001/0033656, hereinafter "Gligor") in view of Jutla (*Encryption Modes with Almost Free Message Integrity*, August 2000, hereinafter "Jutla"), in further view of Menezes, (*Handbook of Applied Cryptography*, 1997, hereinafter "Menezes").

Objection to the Abstract

The abstract was objected to for being too long.

Applicant has supplied a new Abstract that is within the 150 word limit. No new matter has been added.

Objections to the Drawings

The drawings were objected to because FIGs. 6-10 were not marked as "prior art."

Applicant has amended FIGs. 6-10 to include the proper markings. No new matter has been added.

Rejections under 35 U.S.C. §101

Claims 67 and 69 were objected to because they are directed to non-statutory subject matter.

Applicant has amended claims 67 and 69 to incorporate the suggestions of the Examiner.

Rejections under 35 U.S.C. §103(a)

Claims 69-70 were rejected as being unpatentable over Gligor in view of Jutla in further view of Menezes.

Applicant respectfully disagrees that claims 69-70 are obvious in view of Gligor, Jutla, and Menezes. A paper by the Applicant on offset codebook (OCB) encryption was accepted into a competitive conference on cryptography, *The 8th ACM Conference on Computer and Communications Security* (commonly referred to as the ACM CCS conference), a well-regarded venue with an acceptance rate (for the year in question, 2001) of 18%. The paper's only significant contribution is the OCB scheme—the same scheme that is narrowly described by claims 69-70. The OCB scheme is clearly described in the ACM CCS paper as being an improvement to the work of Gligor and Jutla, work that was known in the inventor's community at the time of the ACM CCS submission and was clearly referenced in the paper. If the cryptographic community viewed the subject claims as obvious in view of Gligor and Jutla and Menezes, the paper would most certainly not have been accepted. A copy of the inventor's ACM CCS paper is included with this response.

The inventor's ACM CCS paper above was regarded as one of the top papers at the conference and was therefore invited to a well-respected journal, *ACM Transactions on Information and Systems Security* (ACM TISSEC), where it underwent additional review and subsequently appeared. If the inventor's community viewed the subject claims as obvious in view of Gligor, Jutla, and Menezes, the paper would not have been invited into a well-regarded journal. The inventor's ACM TISSEC paper is included with this response.

Additionally, the commercial community has already decided that OCB is non-obvious: the method has been licensed multiple times, yielding 6-figure earnings, to medium and large companies. The attorneys at these companies know that there is pending IP of Gligor and Jutla (both VDG Inc. and IBM have made public disclosures in connection with activities at NIST, the IEEE, and other

venues) and yet they have chosen to license OCB, never expressing any concern that the patent claims would be seen as obvious given prior work.

The Applicant, knowing intimately the contents of Gligor et al and Jutla and the book by Menezes et al, spent months of intensive effort to develop the disclosed methods, as narrowly defined by the subject claims. Many attempts failed: there were 14 unpublished versions of OCB, many of which had subtle bugs that took weeks to discover by the inventor or the coauthors of his papers. The inventor and his coauthors are of more than ordinary skill in the art; the inventor is a well-known cryptographer with more than 3000 references to his papers, and the winner of the RSA Prize in Mathematics. He did not find the method described by the claims as obvious, nor did co-author Mihir Bellare, who is widely regarded as the top cryptographer of his generation.

Gligor's patent application makes clear that his only idea for handling messages that were not a multiple of the block length was to use padding (e.g., 10* pad the final block to make it a multiple of the block length, and then continue by using the padded message). Gligor's patent application mentions no less than *seven times* that messages that have a length other than the block length are to be padded as necessary to get them to be a multiple of the block length. No other approach is mentioned, and, in fact, padding is the only obvious approach to correctly deal with this issue. It was a first goal of the subject patent application to deal with messages that are not a multiple of the block length by a method smarter than the obvious padding approach: messages not a multiple of the block length are *not* padded in the disclosed technique.

Alternatives to padding that one might initially think of do not work because they break the authenticity protection in subtle ways. This is true for many other potential refinements to the schemes of Jutla and Gligor et al, too. As explained in the ACM CCS paper, "We have found schemes of this sort to be amazingly "fragile"—tweak them a little and they break." Representative examples of such breaks are given the "Definition of the checksum" paragraph

and the "Avoiding pretag collisions" paragraphs on pp. 201-202 of the inventor's ACM CCS paper. Applicant maintains that a method is non-obvious when schemes that are very close to it have subtle errors.

Jutla and Gligor et al provided to NIST the most efficient schemes they knew how to construct. Their proposals are less efficient than OCB according to multiple metrics, as described in the descriptive portion of the subject patent application. It is not reasonable to assume that Jutla or Gligor themselves regarded as obvious the techniques described in the pending patent application that could have made their proposals more efficient.

There are technical difficulties with the Examiner's reading of claims 69-70 against Gligor, Jutla, and Menezes. In particular, items (h) and (j) on p. 5 of the Examiner's office action are not analogous to Menezes p. 340, while item (k) is not analogous to Gligor [0025]. Regarding (h) and (j), the examiner's refers to the Matyas-Meyer-Oseas hash function, a classical method to construct a hash function from a block cipher. There is no length-encoding used in this hash function, and there is no xoring of a portion of a block cipher output with a string having length possibly less than the length of the block cipher. It is simply a chaining method, like CBC encryption. As for item (k), Gligor [0025] mentions padding in an unrelated context—as a way to ensure that all messages acted on in Gligor's scheme have length that is a multiple of n bits (denoted as "I" bits by Gligor). Such padding aims to solve a problem solved by the current patent application—handling messages that are not a multiple of the block size—but it does so at a cost of longer ciphertexts. The padding in (k) of the patent application is taking a fragment (something less than n bits) and adding bits (e.g. 0-bits) simply in order to feed it into the checksum calculation—a checksum calculation that would not work correctly if, for example, a padded message fragment were directly used. This is unrelated to Gligor's initial padding of the input message where he aims to avoid otherwise dealing with "peculiar-length" strings.

Finally, Applicant comments that in providing a block-cipher-based cryptographic mode of operation (whether for encryption, message authentication, authenticated encryption, collision-intractable hashing, or some other end), anything one does is going to be a combining of basic building blocks (apply the block cipher, xor the following words, partition a message, add some padding, combine the following strings, and so forth). The non-obvious part is finding the right way to combine basic operational elements in order to more simply and efficiently accomplish some cryptographic task. In this domain, specious approaches are common and finding a correct way to meld basic building blocks can be highly non-obvious.

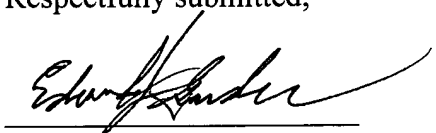
Applicant has amended claims 69 and 70 to clarify that the present invention encrypts messages of arbitrary length into a ciphertext of the same length. These amendments find support on page 24, line 32 to page 25, line 7 of the instant application.

Hence, Applicant respectfully submits that independent claims 67-70 are in condition for allowance.

CONCLUSION

It is submitted that the present application is presently in form for allowance. Such action is respectfully requested.

Respectfully submitted,

By 

Edward J. Grundler
Registration No. 47,615

Date: 6 October 2005

Edward J. Grundler
PARK, VAUGHAN & FLEMING LLP
2820 Fifth Street
Davis, CA 95616-7759
Tel: (530) 759-1663
FAX: (530) 759-1665
Email: edward@parklegal.com

OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption

PHILLIP ROGAWAY
University of California at Davis and Chiang Mai University
MIHIR BELLARE
University of California at San Diego
and
JOHN BLACK
University of Colorado at Boulder

We describe a parallelizable block-cipher mode of operation that simultaneously provides privacy and authenticity. OCB encrypts-and-authenticates a nonempty string $M \in \{0,1\}^*$ using $\lceil |M|/n \rceil + 2$ block-cipher invocations, where n is the block length of the underlying block cipher. Additional overhead is small. OCB refines a scheme, IAPM, suggested by Charanjit Jutla. Desirable properties of OCB include the ability to encrypt a bit string of arbitrary length into a ciphertext of minimal length, cheap offset calculations, cheap key setup, a single underlying cryptographic key, no extended-precision addition, a nearly optimal number of block-cipher calls, and no requirement for a random IV. We prove OCB secure, quantifying the adversary's ability to violate the mode's privacy or authenticity in terms of the quality of its block cipher as a pseudorandom permutation (PRP) or as a strong PRP, respectively.

Categories and Subject Descriptors: E.3 [**Data Encryption**]: Standards

General Terms: Security, Performance, Theory

Additional Key Words and Phrases: AES, authenticity, block-cipher usage, cryptography, encryption, integrity, modes of operation, provable security, standards

---

An earlier version of this paper appears as Rogaway et al. [2001a].
Mihir Bellare received support from NSF grant CCR-0098123, NSF grant ANR-0129617, and an IBM Faculty Partnership Development Award.
John Black received support from NSF CAREER award CCR-0133985 and the University of Colorado. Part of this work was carried out while J. Black was at the University of Nevada, Reno. This work was carried out while P. Rogaway was on leave of absence from UC Davis, visiting the Department of Computer Science, Faculty of Science, Chiang Mai University.
Authors' addresses: Phillip Rogaway, Department of Computer Science, Engineering II Building, University of California, Davis, CA 95616; and Department of Computer Science, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand; email: rogaway@cs.ucdavis.edu, web: www.cs.ucdavis.edu/~rogaway; Mihir Bellare, Department of Computer Science and Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093; email: mihir@cs.ucsd.edu, web: www-cse.ucsd.edu/users/mihir; John Black, Department of Computer Science, 430 UCB, University of Colorado, Boulder, CO 80309; email: jrblack@cs.colorado.edu, web: www.cs.colorado.edu/~jrblack.

1. INTRODUCTION

Background. An authenticated-encryption scheme is a shared-key encryption scheme whose goal is to provide both privacy and authenticity. The encryption algorithm takes a key, a plaintext, and an initialization vector (IV), and it returns a ciphertext. The decryption algorithm takes a key, a ciphertext, and an IV, and it returns either a plaintext or a special symbol, INVALID. We refer to the IV as the *nonce*, to reflect the requirements we will make on it. In addition to the customary privacy goal, an authenticated-encryption scheme aims for authenticity: if an adversary should try to create some new ciphertext, the decryption algorithm will almost certainly regard it as INVALID.

An authenticated-encryption scheme can be constructed by appropriately combining an encryption scheme and a message authentication code (MAC), an approach used pervasively in practice and in standards. (Analyses of such methods are provided in Bellare and Namprempre [2000] and Krawczyk [2001].) But an attractive and long-standing goal has been an authenticated-encryption scheme having computational cost significantly lower than the cost to encrypt plus the cost to MAC. The classical approach for trying to do this is to encrypt-with-redundancy, where one appends a noncryptographic checksum to the message before encrypting it, typically with CBC mode. Many such schemes have been broken. Recently, however, Charanjit Jutla has proposed two authenticated-encryption schemes supported by a claim of provable security [Jutla 2001a]; Gligor and Donescu [2002] then described a different authenticated-encryption scheme. We continue in this line of work. See the Appendix for further history.

OCB Mode. This paper describes a new mode of operation, OCB, which refines one of Jutla's schemes, IAPM [Jutla 2001a]. OCB (which stands for "offset codebook") retains the principal characteristics of IAPM: it is fully parallelizable and adds minor overhead compared to conventional, privacy-only modes. But OCB combines the following additional features:

—*Arbitrary-length messages + minimal-length ciphertexts.* Any string $M \in \{0, 1\}^*$ can be encrypted; in particular, $|M|$ need not be a multiple of the block length n. What is more, the resulting ciphertexts are as short as possible.

—*Minimal IV requirements.* Like other encryption modes, OCB requires an IV. The entity that encrypts chooses a new IV for every message with the only restriction that no IV is used twice. We henceforth refer to the IV as the *nonce.*

—*Nearly optimal number of block-cipher calls.* OCB uses $\lceil |M|/n \rceil + 2$ block-cipher invocations to encrypt-and-authenticate a nonempty message M.

—*Single underlying key.* The key used for OCB is a single block-cipher key, and all block-cipher invocations are keyed by this one key.

—*Efficient offset calculations.* As with other recent methods, we require a sequence of *offsets.* We generate them in a particularly cheap way, each offset requiring a few machine cycles and no extended-precision arithmetic.

Achieving the properties above requires putting together a variety of "tricks" that work together in just the right way. Many plausible-looking constructions that we considered turned out to be wrong.

Performance. Experiments by Lipmaa [2001] on a Pentium III processor show that OCB is about 6.5% slower than the privacy-only mode CBC, and about 54% the speed of CBC encryption combined with the CBC MAC. These figures assume a block cipher of AES128 [US National Institute of Standards 2001].

In settings where there is adequate opportunity for parallelism, OCB will be faster than CBC. Parallelizability is important for obtaining the highest speeds from special-purpose hardware, and it may become useful on commodity processors. For special-purpose hardware, one may want to encrypt-and-authenticate at speeds near 10 Gb/s—an impossible task, with today's technology, for modes like CBC encryption and the CBC MAC. (One could always create a mode that interleaves message blocks fed into separate CBC encryption or CBC MAC calculations, but that would be a new mode, and one with many drawbacks.) For commodity processors, there is an architectural trend toward highly pipelined machines with multiple instruction pipes and lots of registers. Optimally exploiting such features necessitates algorithms that have plenty to do in parallel.

Security Properties. We prove OCB secure, in the sense of reduction-based cryptography. Specifically, we prove indistinguishability under chosen-plaintext attack [Bellare et al. 1997; Goldwasser and Micali 1984] and authenticity of ciphertexts [Bellare and Namprempre 2000; Bellare and Rogaway 2000; Katz and Yung 2000b]. This combination implies indistinguishability under the strongest form of chosen-ciphertext attack (CCA) and that, in turn, is equivalent to nonmalleability under CCA [Bellare et al. 1998; Bellare and Namprempre 2000; Dolev et al. 2000; Katz and Yung 2000a, 2000b]. (Nonmalleability refers to an adversary's inability to modify a ciphertext in a way that makes related the two underlying plaintexts.) Our proof of privacy assumes that the underlying block cipher is good in the sense of a pseudorandom permutation (PRP) [Bellare et al. 2000; Luby and Rackoff 1988], while our proof of authenticity assumes that the block cipher is a strong PRP [Luby and Rackoff 1988]. Our results are quantitative; the security analysis is in the concrete-security paradigm.

We emphasize that OCB has stronger security properties than standard modes. In particular, nonmalleability and indistinguishability under CCA are not achieved by CBC, or by any other standard mode, but these properties are achieved by OCB. We believe that the lack of strong security properties has been a problem for the standard modes of operation, because many users of encryption implicitly assume these properties when designing their protocols. For example, it is common to see protocols that use symmetric encryption in order to "bind together" the parts of a plaintext, or that encrypt related messages as a way to do a "handshake." Standard modes do not support such practices. This fact has sometimes led practitioners to incorrectly apply the standard modes, or to invent or select wrong ways to try to encrypt with authenticity (a well-known example is the use of PCBC mode [Meyer and Matyas 1982] in Kerberos

v.4 [Steiner et al. 1988]). We believe that authenticated-encryption modes are less likely to be misused because many common ways of using a mode of operation that are incorrect when the mode provides privacy only become correct when it provides both privacy and authenticity.

By way of comparison, a chosen-ciphertext attack by Bleichenbacher on the public-key encryption scheme of RSA PKCS #1, v.1.5, motivated the company that controls this de facto standard to promptly upgrade its scheme [Bleichenbacher 1998; RSA Laboratories 1998]. In this public-key setting, it was even a concern if misimplementations could lead to effective attack [Manger 2001]. In contrast, people seem to accept as a matter of course symmetric encryption schemes that are not even nonmalleable. This may be changing, as it becomes clear how damaging and widespread side-channel and chosen-ciphertext attacks can be [Black and Urtubia 2002; Vaudenay 2002].

2. PRELIMINARIES

Notation. If a and b are integers, $a \le b$, then $[a..b]$ is the set $\{a, a+1, \ldots, b\}$. If $i \ge 1$ is an integer then $\mathsf{ntz}(i)$ is the number of trailing 0-bits in the binary representation of i (equivalently, $\mathsf{ntz}(i)$ is the largest integer z such that 2^z divides i). So, for example, $\mathsf{ntz}(7) = 0$ and $\mathsf{ntz}(8) = 3$.

A *string* is a finite sequence of symbols, each symbol being 0 or 1. The string of length 0 is called the *empty string* and is denoted ε. Let $\{0,1\}^*$ denote the set of all strings. If $A, B \in \{0,1\}^*$ then AB, or $A\|B$, is their concatenation. If $A \in \{0,1\}^*$ and $A \neq \varepsilon$ then $\mathsf{firstbit}(A)$ is the first bit of A and $\mathsf{lastbit}(A)$ is the last bit of A. Let i, n be nonnegative integers. Then 0^i and 1^i denote the strings of i 0s and 1s, respectively. Let $\{0,1\}^n$ denote the set of all strings of length n. If $A \in \{0,1\}^*$ then $|A|$ denotes the length of A, in bits, while $\|A\|_n = \max\{1, \lceil |A|/n \rceil\}$ denotes the length of A in n-bit blocks, where the empty string counts as one block. For $A \in \{0,1\}^*$ and $|A| \le n$, $\mathsf{zpad}_n(A)$ is the string $A\,0^{n-|A|}$. With n understood we will write $A\,0^*$ for $\mathsf{zpad}_n(A)$. If $A \in \{0,1\}^*$ and $\tau \in [0..|A|]$ then A [first τ bits] and A [last τ bits] denote the first τ bits of A and the last τ bits of A, respectively. Both of these values are the empty string if $\tau = 0$. If $A, B \in \{0,1\}^*$ then $A \oplus B$ is the bitwise xor of A [first ℓ bits] and B [first ℓ bits], where $\ell = \min\{|A|, |B|\}$ (where $\varepsilon \oplus A = A \oplus \varepsilon = \varepsilon$). So, for example, $1001 \oplus 11 = 01$. If $A = a_{n-1} \cdots a_1 a_0 \in \{0,1\}^n$ then $\mathsf{str2num}(A)$ is the number $\sum_{i=0}^{n-1} 2^i a_i$. If $a \ge 0$ is a number then $\mathsf{num2str}_n(a)$ is the n-bit string A such that $\mathsf{str2num}(A) = a$. Let $\mathsf{len}_n(A) = \mathsf{num2str}_n(|A| \bmod 2^n)$. We omit the subscript when n is understood. Note that if $|A| \ge 2^n$ (which, in practice, will never happen) then $\mathsf{len}_n(A)$ does not encode all of $|A|$ (since we do not have enough bits).

If $A = a_{n-1}a_{n-2} \cdots a_1 a_0 \in \{0,1\}^n$ then the n-bit string $a_{n-2}a_{n-3} \cdots a_1 a_0 0$, denoted $A \ll 1$, is a left shift of A by 1 bit (the first bit of A disappearing and a zero coming into the last bit), while $A \gg 1$ is the n-bit string $0a_{n-1}a_{n-2} \cdots a_2 a_1$ that is a right shift of A by 1 bit (the last bit disappearing and a zero coming into the first bit).

In pseudocode we write "Partition M into $M[1] \cdots M[m]$" as shorthand for "let $m = \|M\|_n$ and let $M[1], \ldots, M[m]$ be strings such that $M[1] \cdots M[m] = M$ and $|M[i]| = n$ for $1 \le i < m$." We write "Partition C into $C[1] \cdots C[m]$ T"

as shorthand for "if $|C| < \tau$ then return INVALID. Otherwise, let $C = C$ [first $|C| - \tau$ bits], let $T = C$ [last τ bits], let $m = \|C\|_n$, and let $C[1], \ldots, C[m]$ be strings such that $C[1] \cdots C[m] = C$ and $|C[i]| = n$ for $1 \le i < m$." Recall that $\|M\|_n = \max\{1, \lceil |M|/n \rceil\}$, so the empty string partitions into $m = 1$ block, that one block being the empty string.

The Field with 2^n Points. Let GF(2^n) denote the field with 2^n points. We interchangeably think of a point a in GF(2^n) in any of the following ways: (1) as an abstract point in a field; (2) as an n-bit string $a_{n-1} \cdots a_1 a_0 \in \{0,1\}^n$; (3) as a formal polynomial $a(\mathrm{x}) = a_{n-1}\mathrm{x}^{n-1} + \cdots + a_1\mathrm{x} + a_0$ with binary coefficients; (4) as an integer between 0 and $2^n - 1$, where the string $a \in \{0,1\}^n$ corresponds to the number str2num(a). For example, one can regard the string $a = 0^{125}101$ as a 128-bit string, as the number 5, as the polynomial $\mathrm{x}^2 + 1$, or as an abstract point in GF(2^{128}). We write $a(\mathrm{x})$ instead of a if we wish to emphasize that we are thinking of a as a polynomial.

To add two points in GF(2^n), take their bitwise xor. We denote this operation by $a \oplus b$. To multiply two points in the field, first fix an irreducible polynomial $p_n(\mathrm{x})$ having binary coefficients and degree n: say the lexicographically first polynomial among the irreducible degree n polynomials having a minimum number of nonzero coefficients. For $n = 128$, the indicated polynomial is $p_{128}(\mathrm{x}) = \mathrm{x}^{128} + \mathrm{x}^7 + \mathrm{x}^2 + \mathrm{x} + 1$. Some other $p_n(\mathrm{x})$-values are $\mathrm{x}^{64} + \mathrm{x}^4 + \mathrm{x}^3 + \mathrm{x} + 1$ (for $n = 64$) and $\mathrm{x}^{256} + \mathrm{x}^{10} + \mathrm{x}^5 + \mathrm{x}^2 + 1$ (for $n = 256$). To multiply $a, b \in$ GF(2^n), which we denote $a \cdot b$, regard a and b as polynomials $a(\mathrm{x}) = a_{n-1}\mathrm{x}^{n-1} + \cdots + a_1\mathrm{x} + a_0$ and $b(\mathrm{x}) = b_{n-1}\mathrm{x}^{n-1} + \cdots + b_1\mathrm{x} + b_0$, form their product $c(\mathrm{x})$ over GF(2), and take the remainder one gets when dividing $c(\mathrm{x})$ by $p_n(\mathrm{x})$.

It is computationally simple to multiply $a \in \{0,1\}^n$ by x. We illustrate the method for $n = 128$, in which case multiplying $a = a_{n-1} \cdots a_1 a_0$ by x yields $a_{n-1}\mathrm{x}^n + a_{n-2}\mathrm{x}^{n-1} + a_1\mathrm{x}^2 + a_0\mathrm{x}$. Thus, if the first bit of a is 0, then $a \cdot \mathrm{x} = a \ll 1$. If the first bit of a is 1 then we must add x^{128} to $a \ll 1$. Since $p_{128}(\mathrm{x}) = \mathrm{x}^{128} + \mathrm{x}^7 + \mathrm{x}^2 + \mathrm{x} + 1 = 0$ we know that $\mathrm{x}^{128} = \mathrm{x}^7 + \mathrm{x}^2 + \mathrm{x} + 1$, so adding x^{128} means to xor by $0^{120}10000111$. In summary, when $n = 128$,

$$a \cdot \mathrm{x} = \begin{cases} a \ll 1 & \text{if} \quad \mathsf{firstbit}(a) = 0 \\ (a \ll 1) \oplus 0^{120}10000111 & \text{if} \quad \mathsf{firstbit}(a) = 1. \end{cases}$$

It is similarly easy to divide $a \in \{0,1\}^{128}$ by x (i.e., to multiply a by the multiplicative inverse of x). If the last bit of a is 0, then $a \cdot \mathrm{x}^{-1}$ is $a \gg 1$. If the last bit of a is 1 then we must add (xor) to $a \gg 1$ the value x^{-1}. Since $\mathrm{x}^{128} = \mathrm{x}^7 + \mathrm{x}^2 + \mathrm{x} + 1$ we have that $\mathrm{x}^{-1} = \mathrm{x}^{127} + \mathrm{x}^6 + \mathrm{x} + 1 = 10^{120}1000011$. In summary, when $n = 128$,

$$a \cdot \mathrm{x}^{-1} = \begin{cases} a \gg 1 & \text{if} \quad \mathsf{lastbit}(a) = 0 \\ (a \gg 1) \oplus 10^{120}1000011 & \text{if} \quad \mathsf{lastbit}(a) = 1. \end{cases}$$

Note that the point $huge = \mathrm{x}^{-1}$ is a large number (when viewed as such); in particular, it starts with a 1 bit, so $huge \ge 2^{n-1}$.

If $L \in \{0,1\}^n$ and $i \geq -1$, we write $L(i)$ as shorthand for $L \cdot x^i$. Using the equations just given, we have an easy way to compute from L the values $L(-1)$, $L(0)$, $L(1)$, ..., $L(\mu)$, where μ is a small number.

Gray Codes. For $\ell \geq 1$, a Gray code is an ordering $\gamma^\ell = (\gamma_0^\ell \quad \gamma_1^\ell \ldots \gamma_{2^\ell-1}^\ell)$ of $\{0,1\}^\ell$ such that successive points differ (in the Hamming sense) by just 1 bit. For n a fixed number, OCB makes use of the "canonical" Gray code $\gamma = \gamma^n$ constructed by $\gamma^1 = (0\ 1)$ and, for $\ell > 0$,

$$\gamma^{\ell+1} = \left(0\gamma_0^\ell \quad 0\gamma_1^\ell \cdots 0\gamma_{2^\ell-2}^\ell \quad 0\gamma_{2^\ell-1}^\ell \quad 1\gamma_{2^\ell-1}^\ell \quad 1\gamma_{2^\ell-2}^\ell \cdots 1\gamma_1^\ell \quad 1\gamma_0^\ell\right).$$

It is easy to see that γ is a Gray code. What is more, for $1 \leq i \leq 2^n - 1$, $\gamma_i = \gamma_{i-1} \oplus (0^{n-1}1 \ll \mathsf{ntz}(i))$. This makes it easy to compute successive points.

We emphasize these characteristics of the Gray-code values $\gamma_1, \gamma_2, \ldots, \gamma_{2^n-1}$: that they are distinct and different from 0; that $\gamma_1 = 1$; and that $\gamma_i < 2i$.

Let $L \in \{0,1\}^n$ and consider the problem of successively forming the strings $\gamma_1 \cdot L, \gamma_2 \cdot L, \gamma_3 \cdot L, \ldots, \gamma_m \cdot L$. Of course $\gamma_1 \cdot L = 1 \cdot L = L$. Now, for $i \geq 2$, assume one has already produced $\gamma_{i-1} \cdot L$. Since $\gamma_i = \gamma_{i-1} \oplus (0^{n-1}1 \ll \mathsf{ntz}(i))$, we know that

$$\begin{aligned}
\gamma_i \cdot L &= (\gamma_{i-1} \oplus (0^{n-1}1 \ll \mathsf{ntz}(i))) \cdot L \\
&= (\gamma_{i-1} \cdot L) \oplus (0^{n-1}1 \ll \mathsf{ntz}(i)) \cdot L \\
&= (\gamma_{i-1} \cdot L) \oplus \left(L \cdot x^{\mathsf{ntz}(i)}\right) \\
&= (\gamma_{i-1} \cdot L) \oplus L(\mathsf{ntz}(i)).
\end{aligned}$$

That is, the ith word in the sequence $\gamma_1 \cdot L, \gamma_2 \cdot L, \gamma_3 \cdot L, \ldots$ is obtained by xoring the previous word with $L(\mathsf{ntz}(i))$. Had the sequence we were considering been $\gamma_1 \cdot L \oplus R, \gamma_2 \cdot L \oplus R, \gamma_3 \cdot L \oplus R, \ldots$, the ith word would be formed in the same way for $i \geq 2$, but the first word in the sequence would have been $L \oplus R$ instead of L.

3. THE SCHEME

Parameters. To use OCB one must specify a block cipher and a tag length. The *block cipher* is a function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, where each $E(K, \cdot) = E_K(\cdot)$ is a permutation on $\{0,1\}^n$. Here $\mathcal{K}$ is the set of possible keys (a finite nonempty set) and n is the block length. We insist that $n \geq 64$ and discourage $n < 128$. The *tag length* is an integer $\tau \in [0..n]$. By trivial means, the adversary will be able to forge a valid ciphertext with probability $2^{-\tau}$. The popular block cipher to use with OCB is likely to be AES [US National Institute of Standards 2001]. As for the tag length, a suggested default of $\tau = 64$ is reasonable. Tags of 32 bits are standard in retail banking. Tags of 96 bits are used in IPSec. Using a tag of more than 80 bits adds questionable security benefit, though it does lengthen each ciphertext.

We let OCB-E denote the OCB mode of operation using block cipher E and an unspecified tag length. We let OCB$[E, \tau]$ denote the OCB mode of operation using block cipher E and tag length τ.

Nonces. Encryption under OCB mode requires an n-bit nonce, N. The nonce would typically be a counter (maintained by the sender) or a random value (selected by the sender). Security is maintained even if the adversary can control the nonce, subject to the constraint that no nonce may be repeated within the current session (that is, during the period of use of the current encryption key). The nonce need not be random, unpredictable, or secret.

The nonce N is needed both to encrypt and to decrypt. Typically it would be communicated, in the clear, along with the ciphertext. However, it is out-of-scope how the nonce is communicated to the party who will decrypt. In particular, we do not regard the nonce as part of the ciphertext.

Definition of the Mode. See Figure 1 for a definition and illustration of OCB. The figure defines OCB encryption and decryption. The key space for OCB is the key space $\mathcal{K}$ for the underlying block cipher E.

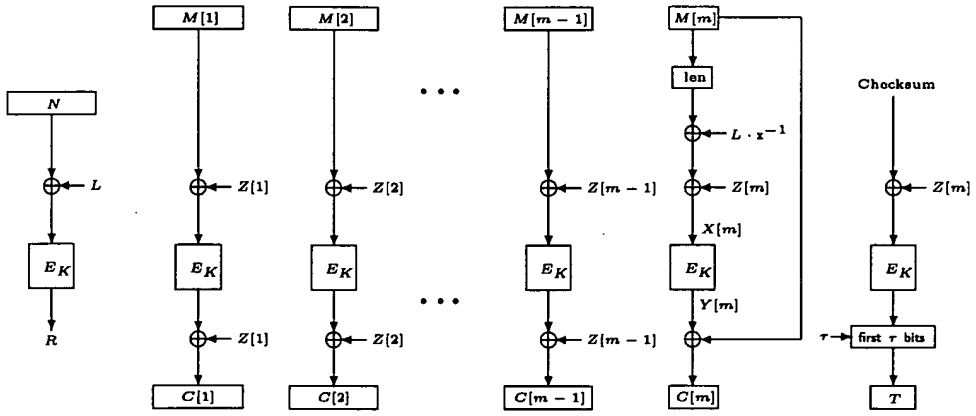
An Equivalent Description. The following description may clarify what a typical implementation might do.

(1) *Key Generation*: Choose a random key $K \xleftarrow{\$} \mathcal{K}$ for the block cipher. The key K is provided to both the entity that encrypts and the entity that decrypts.

(2) *Key Setup*: For the party that encrypts, do any key setup associated with block-cipher enciphering. For the party that decrypts, do any key setup associated with block-cipher enciphering and deciphering. Let $L \leftarrow E_K(0^n)$. Let m bound the maximum number of n-bit blocks that any message that will be encrypted or decrypted may have. Let $\mu \leftarrow \lceil \log_2 m \rceil$. Let $L(0) \leftarrow L$ and, for $i \in [1..\mu]$, compute $L(i) \leftarrow L(i-1) \cdot \mathrm{x}$ using a shift and a conditional xor, as described in Section 2. Compute $L(-1) \leftarrow L \cdot \mathrm{x}^{-1}$ using a shift and a conditional xor, as described in Section 2. Save the values $L(-1)$, $L(0)$, $L(1), \ldots, L(\mu)$ in a table.

(3) *Encryption*: To encrypt plaintext $M \in \{0,1\}^*$ using key K and nonce $N \in \{0,1\}^n$, obtaining a ciphertext C, do the following. Let $m \leftarrow \lceil |M|/n \rceil$. If $m = 0$ then let $m \leftarrow 1$. Let $M[1], \ldots, M[m]$ be strings such that $M[1] \cdots M[m] = M$ and $|M[i]| = n$ for $i \in [1..m-1]$. Let Offset $\leftarrow E_K(N \oplus L)$. Let Checksum $\leftarrow 0^n$. For $i \leftarrow 1$ to $m-1$, do the following: let Checksum $\leftarrow$ Checksum $\oplus$ $M[i]$; let Offset $\leftarrow$ Offset $\oplus$ $L(\mathrm{ntz}(i))$; let $C[i] \leftarrow E_K(M[i] \oplus \mathrm{Offset}) \oplus \mathrm{Offset}$. Now let Offset $\leftarrow$ Offset $\oplus$ $L(\mathrm{ntz}(m))$. Let $Y[m] \leftarrow E_K(\mathrm{len}(M[m]) \oplus L(-1) \oplus \mathrm{Offset})$. Let $C[m] \leftarrow M[m]$ xored with the first $|M[m]|$ bits of $Y[m]$. Let Checksum $\leftarrow$ Checksum $\oplus Y[m] \oplus C[m]\,0^*$. Let T be the first τ bits of $E_K(\mathrm{Checksum} \oplus \mathrm{Offset})$. The ciphertext is $C = C[1] \cdots C[m-1] C[m]\ T$. It must be communicated along with the nonce N.

(4) *Decryption*: To decrypt ciphertext $C \in \{0,1\}^*$ using key K and nonce $N \in \{0,1\}^n$, obtaining a plaintext $M \in \{0,1\}^*$ or an indication INVALID, do the following. If $|C| < \tau$ then return INVALID (the ciphertext has been rejected). Otherwise let C be the first $|C| - \tau$ bits of C and let T be the remaining τ bits. Let $m \leftarrow \lceil |C|/n \rceil$. If $m = 0$ then let $m \leftarrow 1$. Let $C[1], \ldots, C[m]$ be strings



Algorithm $\mathrm{OCB.Enc}_K\ (N, M)$

Partition M into $M[1] \cdots M[m]$
$L \leftarrow E_K(0^n)$
$R \leftarrow E_K(N \oplus L)$
for $i \leftarrow 1$ **to** m **do** $Z[i] = \gamma_i \cdot L \oplus R$
for $i \leftarrow 1$ **to** $m-1$ **do**
  $C[i] \leftarrow E_K(M[i] \oplus Z[i]) \oplus Z[i]$
$X[m] \leftarrow \mathrm{len}(M[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$
$C[m] \leftarrow Y[m] \oplus M[m]$
$C \leftarrow C[1] \cdots C[m]$
Checksum ←
  $M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
$T \leftarrow E_K(\mathrm{Checksum} \oplus Z[m])$ [first τ bits]
return $\mathcal{C} \leftarrow C \,\|\, T$

Algorithm $\mathrm{OCB.Dec}_K\ (N, \mathcal{C})$

Partition $\mathcal{C}$ into $C[1] \cdots C[m]\,T$
$L \leftarrow E_K(0^n)$
$R \leftarrow E_K(N \oplus L)$
for $i \leftarrow 1$ **to** m **do** $Z[i] = \gamma_i \cdot L \oplus R$
for $i \leftarrow 1$ **to** $m-1$ **do**
  $M[i] \leftarrow E_K^{-1}(C[i] \oplus Z[i]) \oplus Z[i]$
$X[m] \leftarrow \mathrm{len}(C[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$
$M[m] \leftarrow Y[m] \oplus C[m]$
$M \leftarrow M[1] \cdots M[m]$
Checksum ←
  $M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
$T' \leftarrow E_K(\mathrm{Checksum} \oplus Z[m])$ [first τ bits]
if $T = T'$ **then return** M
  else return INVALID

Fig. 1. OCB encryption. The message to encrypt is M and the key is K. Message M is written as $M = M[1]M[2] \cdots M[m-1]M[m]$, where $m = \max\{1, \lceil |M|/n \rceil\}$ and $|M[1]| = |M[2]| = \cdots = |M[m-1]| = n$. Nonce N is a nonrepeating value selected by the party that encrypts. It, along with ciphertext $C = C[1]C[2]C[3] \cdots C[m-1]C[m]\ T$, is needed to decrypt. The Checksum is $M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$. Offset $Z[1] = L \oplus R$ while, for $i \geq 2$, $Z[i] = Z[i-1] \oplus L(\mathrm{ntz}(i))$. String L is defined by applying E_K to the fixed string 0^n. For $Y[m] \oplus M[m]$ and $Y[m] \oplus C[m]$, truncate $Y[m]$ if it is longer than the other operand. By $C[m]\,0^*$ we mean $C[m]$ padded on the right with 0-bits to get to length n. Function len represents the length of its argument, mod 2^n, as an n-bit string.

such that $C[1] \cdots C[m] = C$ and $|C[i]| = n$ for $i \in [1..m-1]$. Let Offset $\leftarrow E_K(N \oplus L)$. Let Checksum $\leftarrow 0^n$. For $i \leftarrow 1$ to $m-1$, do the following: let Offset $\leftarrow$ Offset $\oplus\, L(\mathrm{ntz}(i))$; let $M[i] \leftarrow E_K^{-1}(C[i] \oplus \mathrm{Offset}) \oplus \mathrm{Offset}$; let Checksum $\leftarrow$ Checksum $\oplus\, M[i]$. Now let Offset $\leftarrow$ Offset $\oplus\, L(\mathrm{ntz}(m))$. Let $Y[m] \leftarrow E_K(\mathrm{len}(C[m]) \oplus L(-1) \oplus \mathrm{Offset})$. Let $M[m] \leftarrow C[m]$ xored with the first $|C[m]|$ bits of $Y[m]$. Let Checksum $\leftarrow$ Checksum $\oplus\, Y[m] \oplus C[m]\,0^*$. Let T' be the first τ bits of $E_K(\mathrm{Checksum} \oplus \mathrm{Offset})$. If $T \neq T'$ then return

INVALID (the ciphertext has been rejected). Otherwise, the plaintext is $M = M[1] \cdots M[m-1]M[m]$.

4. DISCUSSION

OCB has been designed to have a variety of desirable properties. We mention some of those properties here.

Arbitrary-Length Messages and Minimal Ciphertext Expansion. One of the key characteristics of OCB is that any string $M \in \{0,1\}^*$ can be encrypted, and doing this yields a ciphertext C having $|M| + \tau$ bits. That is, the length of the "ciphertext core"—the portion $C = C[1] \cdots C[m]$ of the ciphertext that excludes the tag—is the same as the length of the message M. This is better, by up to n bits, than what one gets with conventional padding. But remember that we do not regard the nonce as part of the ciphertext. Including it, the amount of information that needs to be sent to the receiver is $|M| + \tau + \eta$ bits, where η bits are used to communicate the nonce N. The value of η could be anything in $[0..n]$, depending on the application.

Single Block-Cipher Key. OCB makes use of just one block-cipher key, K. While $L = E_K(0^n)$ functions rather like a key and would normally be computed at key-setup time, and while standard key-separation techniques can always be used to obtain many keys from one, the point is that, in OCB, all block-cipher invocations use the one key K. Thus only one block-cipher key needs to be setup, saving on storage space and key-setup time.

Weak Nonce Requirements. We believe that modes of operation that require a random IV are often misused. As an example, consider CBC mode, where $C[i] = E_K(M[i] \oplus C[i-1])$ and $C[0] = \text{IV}$. Some standards and books suggest that the IV may be a fixed value, a counter, a timestamp, or the last block of ciphertext from the previous message. But if it is any of these things, one certainly will not achieve any of the standard definitions of privacy [Bellare et al. 1997; Goldwasser and Micali 1984].

It is sometimes suggested that a mode that needs a random IV is preferable to one that needs a nonce: it is said that *state* is needed for a nonce, but not for making random bits. We find this argument wrong. First, a random value of sufficient length can always be used as a nonce, but a nonce cannot be used as a random value. Second, the manner in which systems provide "random" IVs is invariably stateful anyway: unpredictable bits are too expensive to harvest for each IV, so one does this rarely, using state to generate pseudorandom bits from unpredictable bits harvested before. Third, the way to generate pseudorandom bits needs to use cryptography, so the prevalence of noncryptographic pseudorandom number generators routinely results in implementation errors. Fourth, nonce-based schemes facilitate simple replay-detection. Finally, nonces can be communicated using fewer bits than random values.

On-Line. OCB encryption is *on-line*: one can output a stream of ciphertext bits as a stream of plaintext bits arrive, the output stream having constant latency and the transformation using constant memory. When one receives an

indication that the plaintext is over, the final chunk of ciphertext is output. One need not know the length of the plaintext in advance of processing it. This allows the efficient encryption of strings whose representation uses a special character (e.g., a zero-byte) to indicate the string's end. An incremental interface (in the style popular for cryptographic hash functions) could be used to support this functionality.

OCB decryption is likewise on-line, but with an important difference: one can produce a stream of plaintext bits as the stream of ciphertext bits comes in, but when the ciphertext stream is finished one may need to "cancel" the plaintext stream that has issued (having found the ciphertext to be invalid). In such a case, nothing about the ciphertext (such as what was the canceled plaintext) should be adversarially available beyond an indication of its invalidity. In any authenticated-encryption scheme, decryption can be on-line only to this extent.

Significance of Being Efficient. Shaving off a few block-cipher calls or a few bytes of ciphertext may not seem important. But often one is dealing with short messages; for example, roughly a third of the messages on the Internet backbone are 43 bytes. If one is encrypting messages of such short lengths, one should be careful about message expansion and extra computational work since, by percentage, the inefficiencies can be large.

The argument has been made that making a major effort to save a factor of two in computational efficiency is marginal in the first place: "Moore's law" will soon deliver such an improvement anyway, by way of faster hardware. We are not persuaded. Along with processors getting faster, security has become increasingly an issue, and low-power and embedded processors have become more prevalent. The result is a need to cryptographically process more and more data, and often by "dumb" execution vehicles. Hardware advances have changed our understanding of what efficiency entails but, to date, hardware advances have not made cryptographic efficiency less important.

Endian Neutrality. In contrast to a scheme based on mod-p arithmetic (for p a prime near 2^n) or mod-2^n arithmetic, there is almost no endian-favoritism implicit in the definition of OCB. (The exception is that, because of our use of standard mathematical conventions, the left shift used for forming $L(i+1)$ from $L(i)$ is more convenient under a big-endian convention, as is the right shift used for forming $L(-1) = L \cdot \mathrm{x}^{-1}$ from L.)

Optional Preprocessing. Implementations can choose how many $L(i)$ values to precompute. Since only one block-cipher call is needed to compute all of the $L(i)$ values, plus a shift and a conditional xor for each value, it is feasible to do no preprocessing: OCB is appropriate even when each session is a single, short message.

Provable Security. Provable security has become a popular goal for practical protocols. This is because it provides the best way to gain assurance that a cryptographic scheme does what it is should. For a scheme that enjoys provable security one does not need to consider attacks on the scheme, since successful ones imply successful attacks on some simpler object.

When we say that "OCB is provably secure" we are asserting the existence of two theorems. One says that if an adversary A could do a good job at forging ciphertexts with OCB[E, τ] (the adversary does this much more than a $2^{-\tau}$ fraction of the time) then there would be an adversary B that does a good job at distinguishing $(E_K(\cdot), E_K^{-1}(\cdot))$, for a random key K, from $(\pi(\cdot), \pi^{-1}(\cdot))$, for a random permutation $\pi \in \text{Perm}(n)$. The other theorem says that if an adversary A could do a good job at distinguishing OCB[E, τ]-encrypted messages from random strings, then there would be an adversary B that does a good job at distinguishing $E_K(\cdot)$, for a random key K, from $\pi(\cdot)$, for a random permutation $\pi \in \text{Perm}(n)$. Theorems of this sort are called *reductions.* In cryptography, provable security means giving reductions (along with the associated definitions).

Provable security begins with Goldwasser and Micali [1984]. The style of provable security that we use here—where the primitive is a block cipher, the scheme is a mode of operation, and the analysis is concrete (no asymptotics)—is the approach of Bellare and Rogaway [Bellare et al. 1995; 1997; 2000].

It is not enough to know that there is a provable-security result; one should also understand the definitions and the bounds. We have already sketched the definitions. When we speak of the bounds we are addressing "how effective is the adversary B in terms of the efficacy of adversary A" (where A and B are as above). For OCB, the bounds can be roughly summarized as follows: an adversary can always forge with probability $1/2^{\tau}$. Beyond this, the maximal added advantage is at most $\sigma^2/2^n$, where σ is the total number of blocks the adversary sees. The privacy bound likewise degrades as $\sigma^2/2^n$. The conclusion is that one is safe using OCB as long as the underlying block cipher is secure and σ is small compared to $2^{n/2}$. This is the same security degradation one observes for CBC encryption and in the bound for the CBC MAC, as shown by Bellare et al. [1997; 2000]. This kind of security loss was the main motivation for choosing a block length for AES of $n = 128$ bits.

Comparison with Jutla's Bound. More precisely, but still ignoring lower-order terms, our privacy and authenticity bounds are $1.5\sigma^2/2^n$, while Jutla's authenticity bound is insignificantly worse at $2\sigma^2/2^n$ and his privacy bound, rescaled to [0, 1], is insignificantly worse at $3\sigma^2/2^n$ [Jutla 2001b]. Magnifying the latter difference is that the privacy results assume different definitions. Jutla adopts the find-then-guess definition of privacy [Bellare et al. 1997; Goldwasser and Micali 1984], while we use an indistinguishability-from-random-bits definition. The former captures an adversary's inability to distinguish ciphertexts for a pair of adversarially selected, equal-length plaintexts. The latter captures an adversary's inability to distinguish a ciphertext from a random string of the same length. Indistinguishability-from-random-bits implies find-then-guess security, and by a tight reduction, but find-then-guess secure does not imply indistinguishability-from-random-bits. Still, Jutla's scheme probably satisfies the stronger definition, and with similar bounds.

Numerical Example to Illustrate Provable Security. Let us do a small example to illustrate what, concretely, the provable-security results mean. Suppose

that we are using OCB–AES, tags are 64 bits (or longer), and the adversary has access to at most 2^{40} bytes of chosen ciphertext before the key is changed (by a new key-exchange, for example). Then, if AES has its anticipated security, the adversary's chance to produce a valid forged message (after studying its 1 TB of acquired data) is around $1.5\,(2^{40-4})^2/2^{128} + 2^{-64} < 2^{-55}$. In general, the $1.5\sigma^2/2^n$ formula provides guidance in how long a key can be used safely. The considerations are application-dependent, but one needs to change keys well before $2^{n/2}$ blocks have been encrypted.

Simplicity. Simplicity has been a central design goal. Some of OCB's characteristics that contribute to simplicity are (1) short and full final-message-blocks are handled uniformly, not splitting into separate cases; (2) only the simplest form of padding is used: append a minimal number of 0-bits to make a string whose length is a multiple of n. This method is computationally fastest, and helps avoid a proliferation of cases in the analysis; (3) only one algebraic structure is used throughout the algorithm: the finite field GF(2^n); (4) in forming the sequence of offsets, gray-code coefficients are taken monotonically, starting at 1 and stopping at m. One never goes back to an earlier offset or forms more offsets than there are blocks.

Not Fixing How the Nonce is Communicated. We do not specify how the nonce is chosen or communicated. Formally, it is not part of the ciphertext (though the receiving party needs it to decrypt). In many contexts, there is already a natural value to use as a nonce (e.g., a sequence number already present in a protocol flow, or implicit because the parties are communicating over a reliable channel). Even when a protocol is designed from scratch, the number of bits needed to communicate the nonce will vary. In some applications, 32 or even 8 bits is enough. For example, one might have reason to believe that there are at most 2^{32} messages that will flow during the connection, or one may communicate only the lowest 8 bits of a sequence number, counting on the receiver to anticipate the high-order bits.

Not Fixing the Tag Length. The number of bits necessary for the tag vary according to the application. In a context where the adversary obtains something quite valuable from a successful forgery, one may wish to choose a tag length of 80 bits or more. In contexts such as authenticating a video stream, where an adversary would have to forge many frames to have a major impact on the image, an 8-bit tag may be appropriate. With no universally correct value to choose, it is best to leave this parameter unspecified.

Short tags seem to be more appropriate for OCB than for some other MACs, particularly Carter–Wegman MACs. Many Carter–Wegman MACs have the property that if you can forge one message with probability δ then you can forge an arbitrary set of (all correct) messages with probability δ. This does not appear to be true for OCB, though we have not investigated formalizing or proving such properties.

Forming R Using a Block-Cipher Call. During our work we discovered that there are methods for authenticated-encryption that encrypt M using

$\lceil |M|/n \rceil + 1$ block-cipher calls, as opposed to our $\lceil |M|/n \rceil + 2$ calls. Halevi [2001] has also made this finding. However, the methods we know to shave off a block-cipher call either require an unpredictable IV instead of a nonce, or they add conceptual and computational complexity to compute the initial offset R by noncryptographic means (e.g., using a finite-field multiplication of the nonce and a key variant).

Avoiding mod-2^n Addition. Our earlier designs included a scheme based on modular 2^n addition ("addition" for the remainder of this paragraph). Basing an authenticated-encryption scheme on addition is an interesting idea due to Gligor and Donescu [2002]. Compared to our GF(2^n)-based approach ("xor" for the remainder of this paragraph), an addition-based scheme might seem simpler. But the use of addition (where $n \geq 128$) has several disadvantages: (1) The bit-asymmetry of the addition operator implies that the resulting scheme will have a bias toward big-endian architectures or little-endian architectures; there will be no way to achieve an endian neutral scheme. The AES algorithm was constructed to be endian neutral and we wanted OCB–AES to inherit this attribute. (2) Addition is unpleasant for implementations using high-level languages, where one normally has no access to the add-with-carry instruction the machine may have. (3) Addition needs more chip area than xor. (4) Some hardware platforms perform addition more slowly than xor. Experiments on a Pentium 3 revealed that repeated add-with-carry instructions were slower than repeated xors. (5) The concrete security bound appears to be worse with addition than xor (though still not bad). The degradation would seem to be $\Theta(\lg \bar{m})$, where $\bar{m}$ is the maximal message length. We eventually came to believe that the simplicity benefit of addition was not worth it and was not real.

Definition of the Checksum. An initially odd-looking aspect of OCB's definition is the definition of Checksum $= M[1] \oplus \cdots M[m-1] \oplus C[m]\,0^* \oplus Y[m]$. In Jutla's scheme, where one assumes that all messages are a positive multiple of the block length, the checksum is the simpler-looking $M[1] \oplus \cdots M[m-1] \oplus M[m]$. We comment that these two definitions are identical in the case that $|M[m]| = n$. What is more, the definition Checksum $= M[1] \oplus \cdots M[m-1] \oplus M[m]\,0^*$ turns out to be the wrong way to generalize the Checksum to allow for short-final-block messages; in particular, the scheme using that checksum is easily attacked.

5. THEOREMS

5.1 Security Definitions

The first provable-security treatment of symmetric encryption is due to Bellare et al. [1997]. A provable-security treatment of authenticated encryption was initiated by Katz and Yung [2000b] and Bellare and Rogaway [2000] and continued by Bellare and Namprempre [2000]. We build on all these works but our definitions involve some novel elements.

OCB uses a nonce, and we wish to give the adversary every possible advantage (more than is available in real life) by allowing it to choose this nonce (though we forbid the adversary from choosing the same nonce twice). This leads us to introduce a new primitive, that we call a nonce-using symmetric encryption scheme, and that is syntactically different from a standard symmetric encryption scheme. We also introduce a new and particularly strong notion of privacy called indistinguishability from random strings.

Syntax. We extend the syntax of an encryption scheme as given in Bellare et al. [1997]. A (nonce-using, symmetric) encryption scheme is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and an associated number n (the nonce length). Here $\mathcal{K}$ is a finite set and $\mathcal{E}$ and $\mathcal{D}$ are deterministic algorithms. Encryption algorithm $\mathcal{E}$ takes $K \in \mathcal{K}$, $N \in \{0,1\}^n$, and $M \in \{0,1\}^*$, and returns a string $C \leftarrow \mathcal{E}_K(N, M)$. Decryption algorithm $\mathcal{D}$ takes $K \in \mathcal{K}$, $N \in \{0,1\}^n$, and $C \in \{0,1\}^*$, and returns $\mathcal{D}_K(N, M)$, which is either a string $M \in \{0,1\}^*$ or the distinguished symbol INVALID. If $C \leftarrow \mathcal{E}_K(N, M)$ then $\mathcal{D}_K(N, C) = M$.

Privacy. We give a particularly strong definition of privacy, one asserting indistinguishability from random strings. This notion is easily seen to imply (the natural extension to nonce-using schemes) more standard definitions [Bellare et al. 1997], and by tight reductions. Consider an adversary A that has one of two types of oracles: a "real" encryption oracle or a "fake" encryption oracle. A real encryption oracle, $\mathcal{E}_K(\cdot,\cdot)$, takes as input N, M and returns $C \leftarrow \mathcal{E}_K(N, M)$. Assume that $|C| = \ell(|M|)$ depends only on $|M|$. A fake encryption oracle, $\$(\cdot,\cdot)$, takes as input N, M and returns a random string $C \xleftarrow{\$} \{0,1\}^{\ell(|M|)}$. Given adversary A and encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, define $\mathbf{Adv}^{\text{priv}}_{\Pi}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot)} = 1] - \Pr[K \xleftarrow{\$} \mathcal{K} : A^{\$(\cdot,\cdot)} = 1]$.

An adversary A is *nonce-respecting* if it never repeats a nonce: if A asks its oracle a query (N, M) it will never subsequently ask its oracle a query (N, M'), regardless of its coins (if any) and regardless of oracle responses. All adversaries are assumed to be nonce respecting.

Authenticity. We extend the notion of integrity of ciphertexts of Bellare and Namprempre [2000], Bellare and Rogaway [2000] and Katz and Yung [2000b]. Fix an encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and run an adversary A with an oracle $\mathcal{E}_K(\cdot,\cdot)$ for some key K. Adversary A *forges* (in this run) if A is nonce respecting, A outputs (N, C), where $\mathcal{D}_K(N, C) \neq$ INVALID, and A made no earlier query (N, M) that resulted in a response C. Let $\mathbf{Adv}^{\text{auth}}_{\Pi}(A) = \Pr[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot)}$ forges$]$. We stress that the nonce used in the forgery attempt may coincide with a nonce used in one of the adversary's queries.

Block Ciphers and PRFs. A *function family* from n-bits to n-bits is a map $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, where $\mathcal{K}$ is a finite set of strings. It is a *block cipher* if each $E_K(\cdot) = E(K, \cdot)$ is a permutation. Let $\text{Rand}(n)$ denote the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$, and let $\text{Perm}(n)$ denote the set of all permutations from $\{0,1\}^n$ to $\{0,1\}^n$. These sets can be regarded as function families by imagining that each member is specified by a string. For $\pi \in \text{Perm}(n)$, let $\pi^{-1}(Y)$ be

the unique string X such that $\pi(X) = Y$. Let

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{prf}}_{E}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}\colon A^{E_K(\cdot)} = 1] - \Pr[\rho \xleftarrow{\$} \mathrm{Rand}(n)\colon A^{\rho(\cdot)} = 1] \\
\mathbf{Adv}^{\mathrm{prp}}_{E}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}\colon A^{E_K(\cdot)} = 1] - \Pr[\pi \xleftarrow{\$} \mathrm{Perm}(n)\colon A^{\pi(\cdot)} = 1] \\
\mathbf{Adv}^{\mathrm{sprp}}_{E}(A) &= \Pr[K \xleftarrow{\$} \mathcal{K}\colon A^{E_K(\cdot),E_K^{-1}(\cdot)} = 1] \\
&\quad - \Pr\left[\pi \xleftarrow{\$} \mathrm{Perm}(n)\colon A^{\pi(\cdot),\pi^{-1}(\cdot)} = 1\right]
\end{aligned}$$

be defined for a block cipher E and adversary A.

5.2 Theorem Statements

We give information-theoretic bounds on the authenticity and the privacy of OCB. Proofs are in Section 8.

THEOREM 5.1 (AUTHENTICITY). *Fix OCB parameters n and τ. Let A be an adversary that asks q queries and then makes its forgery attempt. Suppose the q queries have aggregate length of σ blocks, and the adversary's forgery attempt has at most c blocks. Let $\bar{\sigma} = \sigma + 2q + 5c + 11$. Then*

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) \le \frac{1.5\,\bar{\sigma}^2}{2^n} + \frac{1}{2^\tau}.$$

The aggregate length of queries $M_1, \ldots, M_q$ means the number $\sigma = \sum_{r=1}^{q} \|M_r\|_n$.

It is standard to pass to a complexity-theoretic analog of Theorem 5.1, but in doing this one will need access to an E^{-1} oracle in order to verify a forgery attempt, which translates into needing the strong PRP assumption. One gets the following. Fix OCB parameters n and τ, and a block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Let A be an adversary that asks q queries and then makes its forgery attempt. Suppose the q queries have aggregate length of σ blocks, and the adversary's forgery attempt has at most c blocks. Let $\bar{\sigma} = \sigma + 2q + 5c + 11$. Let $\delta = \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[E,\tau]}(A) - 1.5\,\bar{\sigma}^2/2^n - 1/2^\tau$. Then there is an adversary B for attacking block cipher E that achieves advantage $\mathbf{Adv}^{\mathrm{sprp}}_{E}(B) \ge \delta$. Adversary B asks at most $q' = \sigma + 2q + c + 3$ oracle queries and has a running time that is equal to A's running time plus the time to compute E or E^{-1} at q' points plus additional time that is $\alpha n \bar{\sigma}$, where the constant α depends only on details of the model of computation.

The privacy of OCB is given by the following result.

THEOREM 5.2 (PRIVACY). *Fix OCB parameters n and τ. Let A be an adversary that asks q queries, these having aggregate length of σ blocks. Let $\bar{\sigma} = \sigma + 2q + 3$. Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) \le \frac{1.5\,\bar{\sigma}^2}{2^n}.$$

As before, it is standard to pass to a complexity-theoretic analog of Theorem 5.2. Fix OCB parameters n and τ, and a block cipher $E\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Let A be an adversary that asks q queries, these having aggregate length

Algorithm	64 B	256 B	1 KB	4 KB
OCB encrypt	24.7 (395)	18.5 (296)	**16.9** (271)	16.7 (267)
ECB encrypt	15.1 (241)	15.0 (239)	**14.9** (238)	14.9 (238)
CBC encrypt	15.9 (254)	15.9 (254)	**15.9** (255)	15.9 (256)
CBC mac	19.2 (307)	16.3 (261)	**15.5** (248)	15.3 (246)

Fig. 2. Performance results from Lipmaa [2001], in cycles per byte (cycles per 16-byte block) on a Pentium III. The block cipher is AES128. Code is written in assembly.

of σ blocks.Let $\bar{\sigma} = \sigma + 2q + 3$. Let $\delta = \mathbf{Adv}^{\text{auth}}_{\text{OCB}[E,\tau]}(A) - 1.5\,\bar{\sigma}^2/2^n$. Then there is an adversary B for attacking block cipher E that achieves advantage $\mathbf{Adv}^{\text{prp}}_E(B) \geq \delta$. Adversary B asks at most $q' = \sigma + 2q + 1$ oracle queries and has a running time that is equal to A's running time plus the time to compute E at q' points plus additional time that is $\alpha n\bar{\sigma}$, where the constant α depends only on details of the model of computation.

6. PERFORMANCE

Abstract Accounting. OCB uses $\lceil |M|/n \rceil + 2$ block-cipher calls to encrypt a nonempty message M. (The empty string takes three block-cipher calls.)

We compare this with CBC encryption and CBC encryption plus a CBC MAC. Namely, "basic" CBC encryption, where one assumes a random IV and a message which is a multiple of the block length, uses $|M|/n$ block-cipher calls. (A more fair comparison uses some padding regime and sets IV $= E_{K'}(N)$, so both schemes use a nonce IV). If one combines basic CBC encryption with a CBC MAC, say MACing the ciphertext (including the IV), then CBC-encryption will use a number of block-cipher calls as just discussed, while the CBC MAC will use at least $\lceil |M|/n \rceil + 1$ block-cipher calls (possibly more, depending on padding conventions and what is done to ensure security across messages of varying lengths). So the total number of calls for CBC encryption with a CBC MAC will be at least $2\lceil |M|/n \rceil + 1$, and typically a bit more.

As with any mode, OCB has overhead beyond the block-cipher calls. Per-block, this overhead is about four n-bit xor operations, plus associated logic. The work for this associated logic will vary according to whether or not one precomputed $L(i)$-values and many additional details.

Though some or all of the needed $L(i)$-values are likely to be precomputed, computing all of them "on the fly" is not inefficient. Starting with 0^n we form successive offsets by xoring the previous offset with L, $2 \cdot L$, L, $4 \cdot L$, L, $2 \cdot L$, L, $8 \cdot L$, and so forth. So half the time we use L itself; a quarter of the time we use $2 \cdot L$; one eighth of the time we use $4 \cdot L$; and so forth. Thus the expected number of times to multiply by x in order to compute an offset is at most $\sum_{i=1}^{\infty} i/2^{i+1} = 1$. Each $a \cdot \mathrm{x}$ instruction requires an n-bit shift and a conditional 32-bit xor. Said differently, for any $m > 0$, the total number of $a \cdot \mathrm{x}$ operations needed to compute $\gamma_1 \cdot L, \gamma_2 \cdot L, \ldots, \gamma_m \cdot L$ is $\sum_{i=1}^{m} \mathrm{ntz}(i)$, which is less than m.

Experimental Results. In Figure 2 we report, with permission, some experimental results by Lipmaa [2001]. On a Pentium III, in optimized assembly, Lipmaa implemented OCB encryption, ECB encryption, CBC encryption, and

the CBC MAC. The last three modes were implemented in their "raw" forms, where one does no padding and assumes that the message acted on is a positive multiple of the block length. For CBC encryption, the IV is fixed. The underlying block cipher is AES128.

Focusing on messages of 1 KB, OCB incurs about 6.4% overhead compared to CBC encryption, and the algorithm takes about 54% of the time of a CBC encryption + CBC MAC. Lipmaa points out that overhead is so low that, in his experiments, an assembly AES128 with a C-code CBC-wrapper is slightly slower than the same AES128 with an assembly OCB-wrapper. Lipmaa's (size-unoptimized) code is 7.2 KB, which includes unrolling AES128 (2.2 KB) three times.

Some aspects of the experiments above are unfavorable to OCB, making the performance estimates conservative. In particular, the "raw" CBC MAC needs to be modified to correctly handle length variability; when combined with CBC encryption, the CBC MAC should be taken over the full ciphertext, including the nonce, which would add an extra block-cipher call; and an extra block-cipher call would normally be performed by CBC to correctly compute the IV from a nonce.

The results above are for a serial execution environment. In settings with plenty of registers and multiple instruction pipes, OCB, properly implemented, will be faster than CBC.

7. AFTERWARDS

After the initial publication of OCB, many individuals pointed out that often times when one is trying to encrypt a message with authenticity there is additional data, such as a message header, that should be authenticated but *not* encrypted. The associated data should be bound to the ciphertext but should not increase its length. This problem of *authenticated encryption with associated data* has been formally defined in Rogaway [2002], and an extension to OCB has been given there that allows the binding-in of arbitrary associated data while retaining OCB's efficiency characteristics.

Schroeppel [2001] has described to us a nice implementation trick that obviates the utility of the Gray-code ordering used in OCB. The method is to precompute the sequence $\hat{L}(0), \hat{L}(1), \hat{L}(2), \hat{L}(3), \ldots = L, 3L, 7L, 15L, \cdots$ instead of $L(0), L(1), L(2), L(3), \ldots = L, 2L, 4L, 8L, \ldots$. Then note that values of the sequence $L, 2L, 3L, 4L, 5L, \ldots$ can be efficiently enumerated using the observation that $iL = (i-1)L \oplus \hat{L}(\mathrm{ntz}(i))$ for any $i \geq 2$.

OCB has become an optional algorithm in a draft IEEE 802.11 standard for the security of wireless LANs.

The authenticated-encryption schemes of Gligor and Donescu [2002], Jutla [2001a], and Rogaway et al. [2001b] all have patents pending. See the first author's web page for current information.

8. PROOFS

8.1 Structure of the Proofs

Our proof of Theorem 5.1 is based on three lemmas. The first, the *structure lemma*, relates the authenticity of OCB to three functions: the M-collision

probability, denoted $\mathrm{Mcoll}_n(\cdot)$, the MM-collision probability, denoted $\mathrm{MMcoll}_n(\cdot,\cdot)$, and the CM-collision probability, denoted $\mathrm{CMcoll}_n(\cdot,\cdot)$. We state this lemma and then explain its purpose and the functions to which it refers.

LEMMA 1 (STRUCTURE LEMMA). *Fix OCB parameters n and τ. Let A be an adversary that asks q queries and then makes its forgery attempt. Suppose the q queries have aggregate length of σ blocks, and the adversary's forgery attempt has at most c blocks. Let $\bar{\sigma} = \sigma + 2q + 5c + 11$. Let $\mathrm{Mcoll}_n(\cdot)$, $\mathrm{MMcoll}_n(\cdot,\cdot)$, and $\mathrm{CMcoll}_n(\cdot,\cdot)$ be the M-, MM-, and CM-collision probabilities, respectively. Then*

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) \le \max_{\substack{m_1,\ldots,m_q \\ \sum_{m_i \ge 1} m_i = \sigma}} \left\{ \sum_{r\in[1..q]} \mathrm{Mcoll}_n(m_r) + \sum_{1\le r<s\le q} \mathrm{MMcoll}_n(m_r, m_s) + \sum_{r\in[1..q]} \mathrm{CMcoll}_n(c, m_r) \right\} + \frac{\bar{\sigma}^2}{2^{n+1}} + \frac{1}{2^{\tau}}.$$

What This Lemma Does. The structure lemma provides a recipe for measuring the maximal forging probability of an adversary attacking the authenticity of OCB: compute the M-, MM-, and CM- collision probabilities, and then put them together using the formula of the lemma.

Informally, $\mathrm{Mcoll}_n(m)$ measures the probability of running into trouble when the adversary asks a single query of the specified length. Trouble means the occurrence of any collision in the associated block-cipher-input values. This includes the "special" input 0^n (used to define $L = E_K(0^n)$ and $N \oplus L$ (used to define $R = E_K(N \oplus L)$). Informally, $\mathrm{MMcoll}_n(m, \bar{m})$ measures the probability of running into trouble when the adversary asks some two oracle queries of the specified lengths. Trouble means that a block-cipher input associated with the first message coincides with a block-cipher input associated with the second message. Informally, $\mathrm{CMcoll}_n(c, \bar{m})$ measures the probability of running into trouble when the adversary tries to forge some particular ciphertext C of the specified block length c, there having been an earlier query of some particular message M of the specified block length m, it receiving some particular response. This time trouble basically refers to the final block-cipher input for the forgery attempt, $X[c+1]$, coinciding with some earlier block-cipher input.

The structure lemma simplifies the analysis of OCB in two ways. First, it allows one to excise adaptivity as a concern. Dealing with adaptivity is a major complicating factor in proofs of this type. Second, it allows one to concentrate on what happens to fixed pairs of messages. It is easier to think about what happens with two messages than what is happening with all $q+1$ of them.

The M- and MM-Collision Probability. We next define the M-collision probability and the MM-collision probability, and then state our upper bound on these functions.

Definition 8.1 (*M- and MM-Collision Probabilities*). Fix $n > 0$ and let $M = M[0] \cdots M[m+1]$ and $\bar{M} = \bar{M}[0] \cdots \bar{M}[\bar{m}+1]$ be strings of at least $2n$ bits, where each $M[i]$ and $\bar{M}[j]$ has n bits. Choose $L, R, \bar{R} \xleftarrow{\$} \{0,1\}^n$ and then associate with M and $\bar{M}$ the points

$$X[-1] = 0^n$$

$$\begin{array}{lll} X[0] = M[0] \oplus L & & \bar{X}[0] = \bar{M}[0] \oplus L \\ X[1] = M[1] \oplus \gamma_1 \cdot L \oplus R & & \bar{X}[1] = \bar{M}[1] \oplus \gamma_1 \cdot L \oplus \bar{R} \\ X[2] = M[2] \oplus \gamma_2 \cdot L \oplus R & & \bar{X}[2] = \bar{M}[2] \oplus \gamma_2 \cdot L \oplus \bar{R} \\ \vdots & & \vdots \\ X[m-1] = M[m-1] \oplus \gamma_{m-1} \cdot L \oplus R & & \bar{X}[\bar{m}-1] = \bar{M}[\bar{m}-1] \oplus \gamma_{\bar{m}-1} \cdot L \oplus \bar{R} \\ X[m] = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R & & \bar{X}[\bar{m}] = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus \bar{R} \\ X[m+1] = M[m+1] \oplus \gamma_{\bar{m}} \cdot L \oplus R & & \bar{X}[\bar{m}+1] = \bar{M}[\bar{m}+1] \oplus \gamma_m \cdot L \oplus \bar{R} \end{array}$$

and the multisets

$$\mathcal{X}_0 = \{X[-1], X[0], X[1], \ldots, X[m], X[m+1]\}$$

$$\mathcal{X} = \{X[0], X[1], \ldots, X[m], X[m+1]\}$$

$$\bar{\mathcal{X}} = \{\bar{X}[0], \bar{X}[1], \ldots, \bar{X}[\bar{m}], \bar{X}[\bar{m}+1]\}$$

Let $\mathrm{Mcoll}_n(M)$ denote the probability that some string is repeated in the multiset $\mathcal{X}_0$, and let $\mathrm{MMcoll}_n(M, \bar{M})$ denote the probability that some element occurs in both $\mathcal{X}$ and $\bar{\mathcal{X}}$. When m and $\bar{m}$ are numbers, let $\mathrm{Mcoll}_n(m)$ denote the maximal value of $\mathrm{Mcoll}_n(M)$ over all strings $M \in (\{0,1\}^n)^{m+2}$, and let $\mathrm{MMcoll}_n(m, \bar{m})$ denote the maximal value of $\mathrm{Mcoll}_n(M, \bar{M})$ over all $M \in (\{0,1\}^n)^{m+2}$ and $\bar{M} \in (\{0,1\}^n)^{\bar{m}+2}$ such that $M[0] \neq \bar{M}[0]$.

Think of $M[0]$ as a synonym for the nonce N, think of $M[m]$ as a generalization of $\mathrm{len}(M[m])$ (where the adversary can effectively control $M[m]$ as opposed to $\mathrm{len}(M[m])$ to influence $X[m]$), and think of $M[m+1]$ as a synonym for Checksum, which we likewise let the adversary control. One similarly understands $\bar{M}[0]$, $\bar{M}[\bar{m}]$, and $\bar{M}[\bar{m}+1]$. The needed bound is as follows.

LEMMA 2 (BOUND ON THE M- AND MM-COLLISION PROBABILITY).

$$\mathrm{Mcoll}_n(m) \leq \binom{m+3}{2} \cdot \frac{1}{2^n} \quad \textit{and} \quad \mathrm{MMcoll}_n(m, \bar{m}) \leq \frac{(m+2)(\bar{m}+2)}{2^n}.$$

The CM-Collision Probability. The CM-collision probability is defined in Figure 3. The following lemma tells us how large it can possibly be.

LEMMA 3 (BOUND ON THE CM-COLLISION PROBABILITY). *Assume* $c, \bar{m} \leq 2^{n-2}$. *Then*

$$\mathrm{CMcoll}_n(c, \bar{m}) \leq \frac{2c + 3\bar{m} + 9}{2^n}.$$

Concluding the Authenticity Theorem. To prove Theorem 5.1, combine Lemmas 1–3. Let $\Pi = \mathrm{OCB}[\mathrm{Perm}(n), \tau]$. Given the aggregate block length σ

```
10  bad ← false;  for all x ∈ {0,1}^n do π(x) ← undefined
11  L ←$ {0,1}^n;  π(0^n) ← L

20  X̄[0] ← N̄ ⊕ L;  Ȳ[0] ← R̄ ←$ {0,1}^n
21  for i ← 1 to m̄ do Z̄(i) ← γ_i · L ⊕ R̄
22  for i ← 1 to m̄ − 1 do { X̄[i] ← M̄[i] ⊕ Z̄[i];  Ȳ[i] ← C̄[i] ⊕ Z̄[i] }
23  X̄[m̄] ← len(M̄[m̄]) ⊕ huge · L ⊕ Z̄[m̄] ;  Ȳ(m̄) ← C̄[m̄] 0* ⊕ M̄[m̄] 0*
24  Checksum' ← M̄[1] ⊕ ··· ⊕ M̄[m̄ − 1] ⊕ C̄[m̄] 0* ⊕ Ȳ[m̄]
25  X̄[m̄ + 1] ← Checksum' ⊕ Z̄[m̄]
26  for i ← 0 to m̄ + 1 do π(X̄[i]) ← Ȳ[i]

30  X[0] ← N ⊕ L
31  if N ≠ N̄ and X[0] ∈ Domain(π) then bad ← true
32  if N = N̄ then R ← R̄ else R ←$ {0,1}^n
33  π(X[0]) ← R
34  for i ← 1 to c do Z[i] ← γ_i · L ⊕ R
35  for i ← 1 to c − 1 do {
36      Y[i] ← C[i] ⊕ Z[i]
37      if Y[i] ∈ Range(π) then X[i] ← π^{-1}(Y[i]) else X[i] ←$ {0,1}^n
38      π(X[i]) ← Y[i];  M[i] ← X[i] ⊕ Z[i] }
39  X[c] ← len(C[c]) ⊕ huge · L ⊕ Z[c]
40  if X[c] ∈ Domain(π) then Y[c] ← π(X[c]) else Y[c] ←$ {0,1}^n
41  π(X[c]) ← Y[c]
42  Checksum ← M[1] ⊕ ··· ⊕ M[c − 1] ⊕ C[c] 0* ⊕ Y[c]
43  X[c + 1] ← Checksum ⊕ Z[c]
44  if X[c + 1] ∈ Domain(π) then bad ← true
```

Fig. 3. Defining the CM-collision probability. The function $\mathrm{CMcoll}_n(\bar{N}, \bar{M}, \bar{C}, N, C)$ is defined as the probability that *bad* gets set to true when executing this game. The value $\mathrm{CMcoll}_n(c, \bar{m})$ is the maximal value of $\mathrm{CMcoll}_n(\bar{N}, \bar{M}, \bar{C}, N, C)$ over all $\bar{m}$-block $\bar{M}$ and $\bar{C}$, and all c-block C such that $N \neq \bar{N}$ or $C \neq \bar{C}$.

and the bound c on the length of the forgery attempt, one must bound the maximum possible value of

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A) &\leq \max_{\substack{m_1,\ldots,m_q \\ \sum_{m_i \geq 1} m_i = \sigma}} \left\{ \sum_{r \in [1..q]} \mathrm{Mcoll}_n(m_r) + \sum_{1 \leq r < s \leq q} \mathrm{MMcoll}_n(m_r, m_s) \right. \\
&\qquad \left. + \sum_{r \in [1..q]} \mathrm{CMcoll}_n(c, m_r) \right\} + \frac{\bar{\sigma}^2}{2^{n+1}} + \frac{1}{2^{\tau}} \\
&\leq \max_{\substack{m_1,\ldots,m_q \\ \sum_{m_i \geq 0} m_i = \sigma}} \left\{ \sum_{r \in [1..q]} \frac{(m_r + 3)^2}{2^{n+1}} + \sum_{1 \leq r < s \leq q} \frac{(m_r + 2)(m_s + 2)}{2^n} \right. \\
&\qquad \left. + \sum_{r \in [1..q]} \left(\frac{2c + 3m_r + 9}{2^n} \right) \right\} + \frac{(\sigma + 2q + 5c + 11)^2}{2^{n+1}} + \frac{1}{2^{\tau}}.
\end{aligned}$$

One can bound the first sum by letting $m_1 = \sigma$ and letting the remaining $m_i = 0$, one can bound the second sum by letting each $m_i = \sigma/q$, and one can bound the third sum by letting $m_1 = \sigma$ and letting the remaining $m_i = 0$. These choices can be justified by the technique of Lagrange multipliers. This gives

$$\begin{aligned}\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A) &\le \frac{0.5(\sigma+3)^2+4.5q}{2^n} + \frac{0.5q^2(\sigma/q+2)^2}{2^n} + \frac{2c+3\sigma+9+q(2c+9)}{2^n} \\ &\quad + \frac{0.5(\sigma+2q+5c+11)^2}{2^n} + \frac{1}{2^\tau} \\ &\le \frac{0.5(\sigma+3)^2+4.5q+0.5(\sigma+2q)^2+2c+3\sigma+9+2cq+9q+0.5(\sigma+2q+5c+11)^2}{2^n} \\ &\quad + \frac{1}{2^\tau} \le \frac{0.5(\sigma+3)^2+0.5(\sigma+2q)^2+0.5(\sigma+2q+5c+11)^2+(3\sigma+2cq+2c+13.5q+9)}{2^n} \\ &\quad + \frac{1}{2^\tau} \le \frac{1.5\,(\sigma+2q+5c+11)^2}{2^n} + \frac{1}{2^\tau} \le \frac{1.5\,\bar{\sigma}^2}{2^n} + \frac{1}{2^\tau}.\end{aligned}$$

The fourth inequality is justified by checking that $0.5(\sigma + 3 + (2q + 5c + 8))^2 - 0.5(\sigma + 3)^2)$ already exceeds $3\sigma + 2cq + 2c + 13.5q + 9$. This completes the proof. □

Privacy. Privacy is obtained rather easily en route to proving authenticity. This is because of the following result, which closely follows the first half of the proof of the structure lemma.

LEMMA 4 [PRIVACY LEMMA]. *Fix OCB block length n and tag length τ, and let $\Pi = \mathrm{OCB}[\mathrm{Perm}(n), \tau]$. Let A be an adversary that asks q queries, these having aggregate block length of σ blocks. Let $\mathrm{Mcoll}_n(\cdot)$ and $\mathrm{MMcoll}_n(\cdot,\cdot)$ be the M- and MM-collision probabilities. Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) \le \frac{(\sigma+2q+1)^2}{2^{n+1}} + \max_{\substack{m_1,\ldots,m_q \\ \sum_{m_i \ge 1} m_i = \sigma}} \left\{ \sum_{r\in[1..q]} \mathrm{Mcoll}_n(m_r) + \sum_{1\le r<s\le q} \mathrm{MMcoll}_n(m_r, m_s) \right\}.$$

Combining Lemmas 2 and 4 gives Theorem 5.2. Namely,

$$\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) \le \frac{(\sigma+2q+1)^2}{2^{n+1}} + \max_{\substack{m_1,\ldots,m_q \\ \sum_{m_i \ge 0} m_i = \sigma}} \left\{ \sum_{r\in[1..q]} \frac{(m_r+3)^2}{2\cdot 2^n} \right\} + \max_{\substack{m_1,\ldots,m_q \\ \sum_{m_i \ge 0} m_i = \sigma}} \left\{ \sum_{1\le r<s\le q} \frac{(m_r+2)(m_s+2)}{2^n} \right\}$$

and we bound the two sums exactly as before, giving

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) &\le \frac{0.5(\sigma+2q+1)^2}{2^n} + \frac{0.5(\sigma+3)^2+4.5q}{2^n} + \frac{0.5q^2(\sigma/q+2)^2}{2^n} \\
&\le \frac{0.5(\sigma+2q+1)^2 + 0.5(\sigma+3)^2 + 4.5q + 0.5(\sigma+2q)^2 + 4.5q}{2^n} \\
&\le \frac{1.5\,(\sigma+2q+3)^2}{2^n} \\
&\le \frac{1.5\,\bar{\sigma}^2}{2^n}.
\end{aligned}$$

The third inequality can be justified by noting that $0.5(\sigma+3+2q)^2 - 0.5(\sigma+3)^2$ exceeds $4.5q$.

8.2 Proof of the Structure Lemma (Lemma 1)

Let A be a (computationally unbounded) adversary that attempts to violate the authenticity of $\Pi = \mathrm{OCB}[\mathrm{Perm}(n), \tau]$. Without loss of generality, A is deterministic. The adversary is given an oracle for $\mathrm{OCB.Enc}_{\pi}(\cdot,\cdot)$. We must bound the probability that A, after adaptively using this oracle q times, on messages with aggregate length σ blocks, produces a properly forged ciphertext having at most c blocks. This forgery probability is denoted $\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A)$.

Game A. One can conceive of A interacting with $\mathrm{OCB.Enc}_{\pi}(\cdot,\cdot)$ and then producing a forgery attempt as A playing a certain game, game A, as defined in Figures 4 and 5. Rather than choose $\pi \xleftarrow{\$} \mathrm{Perm}(n)$ all at once, this game defines the values of $\pi(x)$ point-by-point, as needed. We use the notation $\mathrm{Domain}(\pi)$ for the set of values $x \in \{0,1\}^n$ such that $\pi(x) \neq \mathtt{undefined}$. By $\overline{\mathrm{Domain}}(\pi)$ we mean $\{0,1\}^n \setminus \mathrm{Domain}(\pi)$. Similarly, $\mathrm{Range}(\pi)$ is the set of $y \in \{0,1\}^n$ such that there exists an $x \in \{0,1\}^n$ for which $\pi(x) = y$, and $\overline{\mathrm{Range}}(\pi) = \{0,1\}^n \setminus \mathrm{Range}(\pi)$.

An inspection of game A makes clear that it supplies to A a perfect simulation of $\mathrm{OCB.Enc}_{\pi}(\cdot,\cdot)$. Game A simulates OCB in a somewhat unusual way, not only defining π point-by-point, but, when a value $\pi(x)$ is needed, for some new x, we get this value, in most cases, not by choosing $y \xleftarrow{\$} \overline{\mathrm{Range}}(\pi)$, as would seem natural, but by choosing $y \xleftarrow{\$} \{0,1\}^n$, setting $\pi(x)$ to y if y is not already in the range of π, and "changing our minds," setting $\pi(x) \xleftarrow{\$} \overline{\mathrm{Range}}(\pi)$, otherwise. In the latter case, a flag *bad* is set to `true`. The flag *bad* is also set to `true` when the adversary successfully forges. Consequently, upperbounding the probability that *bad* gets set to `true` in game A serves to upperbound the adversary's forging probability.

Game A′. We begin by making a couple of quite trivial changes to game A. First, instead of setting $C[m] = M[m] \oplus Y[m]$ (in line 24 of game A), we set $C[m] = M[m]\,0^* \oplus Y[m]$, instead. That is, we imagine returning the "full" final-ciphertext-block instead of the truncated final-ciphertext-block. Clearly the extra bits given to the adversary cannot make worse an optimal adversary's chance of successful forgery. Second, instead of returning (in line 30 of game A) a

Initialization:
01 $bad \leftarrow$ **false**; **for all** $x \in \{0,1\}^n$ **do** $\pi(x) \leftarrow$ **undefined**
02 $L \xleftarrow{\$} \{0,1\}^n$; $\pi(0^n) \leftarrow L$

When A **asks query** (N, M): *//q such queries will be asked*
10 Partition M into blocks $M[1] \cdots M[m]$
11 $X[0] \leftarrow N \oplus L$; $Y[0] \xleftarrow{\$} \{0,1\}^n$
12 **if** $X[0] \in \text{Domain}(\pi)$ **then** { $bad \leftarrow$ **true**; $Y[0] \leftarrow \pi(X[0])$ } **else**
13 **if** $Y[0] \in \text{Range}(\pi)$ **then** { $bad \leftarrow$ **true**; $Y[0] \xleftarrow{\$} \overline{\text{Range}}(\pi)$ }
14 $\pi(X[0]) \leftarrow Y[0]$

15 **for** $i \leftarrow 1$ **to** m **do** $Z[i] \leftarrow \gamma_i \cdot L \oplus Y[0]$
16 **for** $i \leftarrow 1$ **to** $m-1$ **do** {
17 $X[i] \leftarrow M[i] \oplus Z[i]$; $Y[i] \xleftarrow{\$} \{0,1\}^n$
18 **if** $X[i] \in \text{Domain}(\pi)$ **then** { $bad \leftarrow$ **true**; $Y[i] \leftarrow \pi(X[i])$ } **else**
19 **if** $Y[i] \in \text{Range}(\pi)$ **then** { $bad \leftarrow$ **true**; $Y[i] \xleftarrow{\$} \overline{\text{Range}}(\pi)$ }
20 $\pi(X[i]) \leftarrow Y[i]$; $C[i] \leftarrow Y[i] \oplus Z[i]$ }

21 $X[m] \leftarrow \text{len}(M[m]) \oplus \mathit{huge} \cdot L \oplus Z[m]$; $Y[m] \xleftarrow{\$} \{0,1\}^n$
22 **if** $X[m] \in \text{Domain}(\pi)$ **then** { $bad \leftarrow$ **true**; $Y[m] \leftarrow \pi(X[m])$ } **else**
23 **if** $Y[m] \in \text{Range}(\pi)$ **then** { $bad \leftarrow$ **true**; $Y[m] \xleftarrow{\$} \overline{\text{Range}}(\pi)$ }
24 $\pi(X[m]) \leftarrow Y[m]$; $C[m] \leftarrow M[m] \oplus Y[m]$

25 $\text{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
26 $X[m+1] \leftarrow \text{Checksum} \oplus Z[m]$; $Y[m+1] \xleftarrow{\$} \{0,1\}^n$
27 **if** $X[m+1] \in \text{Domain}(\pi)$ **then** { $bad \leftarrow$ **true**; $Y[m+1] \leftarrow \pi(X[m+1])$ } **else**
28 **if** $Y[m+1] \in \text{Range}(\pi)$ **then** { $bad \leftarrow$ **true**; $Y[m+1] \xleftarrow{\$} \overline{\text{Range}}(\pi)$ }
29 $\pi(X[m+1]) \leftarrow Y[m+1]$; $T \leftarrow Y[m+1]$ [first τ bits]
30 **return** $\mathcal{C} \leftarrow C[1] \cdots C[m]\,T$

Fig. 4. *Game A*, part 1. This game provides adversary A a perfect simulation of OCB[Perm(n), τ].

tag T which is the first τ bits of $Y[m+1]$, we return the full tag, $Y[m+1]$. Once again, the extra bits provided to the adversary can only improve an optimal adversary's chance of success. Let game A′ denote this new, "easier" game. We will bound the probability that *bad* gets set to `true` in game A′.

Game B. Next we eliminate from game A′ the statement which immediately follows *bad* being set to `true` in each of lines 12, 13, 18, 19, 22, 23, 27, and 28. The **else** statements are also eliminated. This new game, game B, is shown in Figure 6. This new game is different from game A′, and an adversary A having queries answered according to game B will not be seeing the same view as one whose queries are answered according to A′. Still, game B has been constructed so that it behaves identically to game A′ until the flag *bad* is set to `true`. Only at that point do the two games diverge. As a consequence, regardless of the behavior of A, the probability that *bad* will get set to `true` when A plays game B is identical to the probability that *bad* gets set to `true` when A plays game A′. Now we are interested in upperbounding the probability of forgery in

When A makes forgery attempt $(N, \mathcal{C})$:
50 Partition $\mathcal{C}$ into $C[1] \cdots C[c]\, T$
51 $X[0] \leftarrow N \oplus L$; **if** $X[0] \in \text{Domain}(\pi)$ **then** $Y[0] \leftarrow \pi(X[0])$ **else** $Y[0] \xleftarrow{\$} \overline{\text{Range}}(\pi)$
52 $\pi(X[0]) \leftarrow Y[0]$
53 **for** $i \leftarrow 1$ **to** c **do** $Z[i] \leftarrow \gamma_i \cdot L \oplus Y[0]$
54 **for** $i \leftarrow 1$ **to** $c-1$ **do** {
55 $\quad Y[i] \leftarrow C[i] \oplus Z[i]$
56 $\quad$ **if** $Y[i] \in \text{Range}(\pi)$ **then** $X[i] \leftarrow \pi^{-1}(Y[i])$ **else** $X[i] \xleftarrow{\$} \overline{\text{Domain}}(\pi)$
57 $\quad \pi(X[i]) \leftarrow Y[i]$; $M[i] \leftarrow X[i] \oplus Z[i]$ }

58 $X[c] \leftarrow \text{len}(C[c]) \oplus \mathit{huge} \cdot L \oplus Z[c]$
59 **if** $X[c] \in \text{Domain}(\pi)$ **then** $Y[c] \leftarrow \pi(X[c])$ **else** $Y[c] \xleftarrow{\$} \overline{\text{Range}}(\pi)$
60 $\pi(X[c]) \leftarrow Y[c]$
61 $\text{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[c-1] \oplus C[c]\,0^* \oplus Y[c]$
62 $X[c+1] \leftarrow \text{Checksum} \oplus Z[c]$
63 **if** $X[c+1] \in \text{Domain}(\pi)$ **then** $Y[c+1] \leftarrow \pi(X[c])$ **else** $Y[c+1] \xleftarrow{\$} \overline{\text{Range}}(\pi)$
64 $T' \leftarrow Y[c+1]$ [first τ bits]
65 **if** $T = T'$ **then** $\mathit{bad} \leftarrow \texttt{true}$

Fig. 5. Games A, A′ B, B′, and C, part 2.

Initialization:
01 $\mathit{bad} \leftarrow \texttt{false}$; **for all** $x \in \{0,1\}^n$ **do** $\pi(x) \leftarrow \texttt{undefined}$
02 $L \xleftarrow{\$} \{0,1\}^n$; $\pi(0^n) \leftarrow L$

When A asks query (N, M): *//q such queries will be asked*
10 Partition M into blocks $M[1] \cdots M[m]$
11 $X[0] \leftarrow N \oplus L$; $Y[0] \xleftarrow{\$} \{0,1\}^n$
12 **if** $X[0] \in \text{Domain}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
13 **if** $Y[0] \in \text{Range}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
14 $\pi(X[0]) \leftarrow Y[0]$
15 **for** $i \leftarrow 1$ **to** m **do** $Z[i] \leftarrow \gamma_i \cdot L \oplus Y[0]$
16 **for** $i \leftarrow 1$ **to** $m-1$ **do** {
17 $\quad X[i] \leftarrow M[i] \oplus Z[i]$; $Y[i] \xleftarrow{\$} \{0,1\}^n$
18 $\quad$ **if** $X[i] \in \text{Domain}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
19 $\quad$ **if** $Y[i] \in \text{Range}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
20 $\quad \pi(X[i]) \leftarrow Y[i]$; $C[i] \leftarrow Y[i] \oplus Z[i]$ }
21 $X[m] \leftarrow \text{len}(M[m]) \oplus \mathit{huge} \cdot L \oplus Z[m]$; $Y[m] \xleftarrow{\$} \{0,1\}^n$
22 **if** $X[m] \in \text{Domain}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
23 **if** $Y[m] \in \text{Range}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
24 $\pi(X[m]) \leftarrow Y[m]$; $C[m] \leftarrow M[m]\,0^* \oplus Y[m]$
25 $\text{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
26 $X[m+1] \leftarrow \text{Checksum} \oplus Z[m]$; $Y[m+1] \xleftarrow{\$} \{0,1\}^n$
27 **if** $X[m+1] \in \text{Domain}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
28 **if** $Y[m+1] \in \text{Range}(\pi)$ **then** $\mathit{bad} \leftarrow \texttt{true}$
29 $\pi(X[m+1]) \leftarrow Y[m+1]$
30 **return** $\mathcal{C} \leftarrow C[1] \cdots C[m]\, Y[m+1]$

Fig. 6. Game B, part 1.

game A, which we do by upperbounding the probability that *bad* gets set to `true` in game A′, which is just the probability that *bad* gets set to `true` in game B.

Note that we are not claiming that the probability of the adversary forging in game B (meaning that *bad* gets set to `true` at line 65 of game B) is the same as the probability of the adversary forging in A′ (meaning that *bad* gets set to `true` in the last line of that game). Claims of this sort are tempting to make, but they are untrue.

Bounding Y-Collisions in Game B. We next bound the probability that *bad* will be set to `true` in any of lines 13, 19, 23, or 28 of game B. In each of these lines, a random n-bit string was just chosen and then it is tested for membership in the growing set Range(π). In the course of game B, the size Range(π) starts off at 0 and then grows one element at a time until it reaches a final size of $\sigma + 2q + 1$ elements. Therefore, the probability that, in growing Range(π), there is a repetition as we add in random points is at most $(1 + 2 + \cdots + \sigma + 2q)/2^n \leq (\sigma + 2q + 1)^2/2^{n+1}$. We note this for future reference:

$$\Pr[A \text{ causes } bad \text{ to be set in any of lines 13, 19, 23 or 28 of game B}] \leq \frac{(\sigma + 2q + 1)^2}{2^{n+1}}. \quad (1)$$

Having bounded the probability that *bad* will be set in the four indicated lines, we may imagine eliminating these four lines, forming a new game, game B′. The probability that *bad* is set in game B is at most the computed bound more than the probability that *bad* is set in game B′. Thus we may continue the analysis using game B′ as long as we compensate the final bound by adding in the term given by Eq. (1).

Game C. In game B′, consider the distribution on strings returned to the adversary in response to a query (N, M), where $m = \|M\|_n$. The adversary learns $C = C[1] \cdots C[m-1]C[m]\, Y[m+1]$. Since each block of this string is a uniform random value xor'ed with some other, independent value, we have that C is uniformly distributed and independent of the query M, apart from its length. As a consequence, when a query of N, M is made, where M has m blocks, we can return a random answer C (of $nm + n$ bits) and do no more at that time. Later, when the adversary is done making its q queries, we can set the remaining random values, make the associated assignments to π, and set the flag *bad*, as appropriate. This is what has been done in Game C of Figure 7. From the adversary's point of view, game B′ and game C are identical. Furthermore, the probability that *bad* gets set to `true` is identical in the two games.

Game D. We have reduced the problem of upperbounding the forging probability to the problem of upperbounding the probability that *bad* gets set to `true` in game C. This probability is over the coins used in line 11 of game C (which defines the C_r-values) and over the additional coins used subsequently in the program. We must show that, over this sequence of coins (remember that the adversary is deterministic) the flag *bad* is rarely set.

```
When A asks its r-th query, (N_r, M_r):        //r will range from 1 to q
10  Partition M_r into blocks M_r[1] ··· M_r[m_r]
11  C_r[1], ..., C_r[m_r], Y_r[m+1] <-$ {0,1}^n
12  return C_r <- C_r[1] ··· C_r[m_r] Y_r[m_r + 1]

When A is done making oracle queries:
20  bad <- false;  for all x ∈ {0,1}^n do π(x) <- undefined
21  L <-$ {0,1}^n;  π(0^n) <- L

30  for r <- 1 to q do {
31      X_r[0] <- N_r ⊕ L;  Y_r[0] <-$ {0,1}^n
32      for i <- 1 to m_r do Z_r[i] <- γ_i · L ⊕ Y_r[0]
33      for i <- 1 to m_r - 1 do { X_r[i] <- M_r[i] ⊕ Z_r[i];  Y_r[i] <- C_r[i] ⊕ Z_r[i] }
34      X_r[m_r] <- len(M[m_r]) ⊕ huge · L ⊕ Z_r[m_r] ;  Y_r[m_r] <- C_r[m_r] ⊕ M_r[m_r] 0*
35      Checksum_r <- M_r[1] ⊕ ··· ⊕ M_r[m_r - 1] ⊕ C_r[m_r] 0* ⊕ Y_r[m_r]
36      X_r[m_r + 1] <- Checksum_r ⊕ Z_r[m_r] }

37  X <- (X_1[0], X_1[1], ..., X_1[m_1 + 1],  ...,  X_q[0], X_q[1], ..., X_q[m_q + 1])
38  Y <- (Y_1[0], Y_1[1], ..., Y_1[m_1 + 1],  ...,  Y_q[0], Y_q[1], ..., Y_q[m_q + 1])
39  if some string is repeated in X ∪ {0^n} then bad <- true
40  for i <- 1 to |X| do π(X[i]) <- Y[i]
```

Fig. 7. Game C, part 1. This game provides adversary A with the same view as game B, and sets *bad* with the same probability. But it defers some random choices.

We will show something stronger: that even if one fixes all of the coins used in line 11 (the C_r-values) and takes the probability over just the remaining coins, still the probability that *bad* gets set to `true` is small. The virtue of this change is that it effectively eliminates the q interactive queries from the game. Namely, since the adversary A is deterministic and each response C_r has been fixed, the adversary can be imagined to "know" all of the queries $N_1, M_1, \ldots, N_q, M_q$ that it would ask and all of the answers $C_1, \ldots, C_q$ that it would receive. All the adversary has left to do is to output the forgery attempt $(N, C\ T)$. This value too is now predetermined, as our adversary is deterministic. So the adversary is effectively gone, and we are left to claim that for any $N_1, M_1, \ldots, N_q, M_q,\ C_1, \ldots, C_q,\ N, C, T$, the flag *bad* will rarely be set if we run game C starting at line 20. The new game is called game D. It depends on $N_1, M_1, \ldots, N_q, M_q,\ C_1, \ldots, C_q,\ N, C, T$, which are now just constants. The constants are not quite arbitrary: the N_r-values are still required to be distinct. The lengths of $M_1, \ldots, M_q$ are $m_1, \ldots, m_q$ blocks. The length of C is c blocks.

The $Mcoll_n$ and $MMcoll_n$ Terms. At this point we make the observation that *bad* will be set to `true` in line 40 of game D if and only if either

—There is some $r \in [1..q]$ such that there is a repetition in the multiset

$$\{0^n,\ X_r[0],\ X_r[1], \ldots, X_r[m_r]\}$$

—There is some pair $r, s \in [1..q]$, where $r < s$, such that $\{X_r[0], \ldots X_r[m_r + 1]\}$ has some a point in common with $\{X_s[0], \ldots X_s[m_s + 1]\}$.

```
20  bad ← false;  for all x ∈ {0,1}^n do π(x) ← undefined
21  L ←$ {0,1}^n;  π(0^n) ← L

30  for r ← 1 to q do {
31      X_r[0] ← N_r ⊕ L;  Y_r[0] ←$ {0,1}^n
32      for i ← 1 to m_r do Z_r[i] ← γ_i · L ⊕ Y_r[0]
33      for i ← 1 to m_r − 1 do { X_r[i] ← M_r[i] ⊕ Z_r[i];  Y_r[i] ← C_r[i] ⊕ Z_r[i] }
34      X_r[m_r] ← len(M[m_r]) ⊕ huge · L ⊕ Z_r[m_r] ;  Y_r[m_r] ← C_r[m_r] ⊕ M_r[m_r] 0*
35      Checksum_r ← M_r[1] ⊕ ··· ⊕ M_r[m_r − 1] ⊕ C_r[m_r] 0* ⊕ Y_r[m_r]
36      X_r[m_r + 1] ← Checksum_r ⊕ Z_r[m_r] }
37  X ← (X_1[0], X_1[1], ..., X_1[m_1 + 1], ..., X_q[0], X_q[1], ..., X_q[m_q + 1])
38  Y ← (Y_1[0], Y_1[1], ..., Y_1[m_1 + 1], ..., Y_q[0], Y_q[1], ..., Y_q[m_q + 1])
39  for i ← 1 to |X| do π(X[i]) ← Y[i]
40  if some string is repeated in X ∪ {0^n} then bad ← true

50  X[0] ← N ⊕ L; if X[0] ∈ Domain(π) then Y[0] ← π(X[0]) else Y[0] ←$ Range(π)
51  π(X[0]) ← Y[0]
52  for i ← 1 to c do Z[i] ← γ_i · L ⊕ Y[0]
53  for i ← 1 to c − 1 do {
54      Y[i] ← C[i] ⊕ Z[i]
55      if Y[i] ∈ Range(π) then X[i] ← π^{-1}(Y[i]) else X[i] ←$ Domain(π)
56      π(X[i]) ← Y[i];  M[i] ← X[i] ⊕ Z[i] }
57  X[c] ← len(C[c]) ⊕ huge · L ⊕ Z[c]
58  if X[c] ∈ Domain(π) then Y[c] ← π(X[c]) else Y[c] ←$ Range(π)
59  π(X[c]) ← Y[c]
60  Checksum ← M[1] ⊕ ··· ⊕ M[c − 1] ⊕ C[c] 0* ⊕ Y[c]
61  X[c + 1] ← Checksum ⊕ Z[c]
62  if X[c + 1] ∈ Domain(π) then Y[c + 1] ← π(X[c]) else Y[c + 1] ←$ Range(π)
63  T' ← Y[c + 1] [first τ bits]
64  if T = T' then bad ← true
```

Fig. 8. Game D. This game depends on $N_1, \ldots, N_q$, $M_1, \ldots, M_q$, $C_1, \ldots, C_q$, $Y_1[m_1 + 1], \ldots, Y_q[m_q + 1]$, $N, C = C[1] \cdots C[c]$, and T.

The probability of this event is at most

$$\sum_{r \in [1..q]} \mathrm{Mcoll}_n(m_r) + \sum_{1 \le r < s \le q} \mathrm{MMcoll}_n(m_r, m_s) \tag{2}$$

by our definition of Mcoll_n and MMcoll_n. Therefore, the probability that *bad* is set to `true` in line 40 of Game D is at most the expression above. We are left now to focus on the probability that *bad* gets set to `true` in line 64 of Game D (Figures 8 and 5).

Game E. We modify the second half of game D (lines 20–39 are unchanged). First, we simplify lines 50, 55 and 58, and 62 by choosing a random value in $\{0, 1\}^n$ as opposed to a value in the co-range, co-domain, co-range, and co-range of π, respectively. By similar reasoning to that used before, this new game may decrease the probability that *bad* gets set to `true`, but by an amount that is at most

$$\frac{(c+2)(\sigma + 2q + c + 3)}{2^n}.$$

```
50  X[0] ← N ⊕ L
51  if N ≠ N_r for any r and X[0] ∈ Domain(π) then bad ← true
52  if N = N_r for some r then Y[0] ← Y_r[0] else Y[0] ←$ {0,1}^n
53  π(X[0]) ← Y[0]
54  for i ← 1 to c do Z[i] ← γ_i · L ⊕ Y[0]
55  for i ← 1 to c − 1 do {
56        Y[i] ← C[i] ⊕ Z[i]
57        if Y[i] ∈ Range(π) then X[i] ← π^{-1}(Y[i]) else X[i] ←$ {0,1}^n
58        π(X[i]) ← Y[i];   M[i] ← X[i] ⊕ Z[i] }
59  X[c] ← len(C[c]) ⊕ huge · L ⊕ Z[c]
60  if X[c] ∈ Domain(π) then Y[c] ← π(X[c]) else Y[c] ←$ {0,1}^n
61  π(X[c]) ← Y[c]
62  Checksum ← M[1] ⊕ ··· ⊕ M[c − 1] ⊕ C[c]0* ⊕ Y[c]
63  X[c + 1] ← Checksum ⊕ Z[c]
64  if X[c + 1] ∈ Domain(π) then bad ← true
```

Fig. 9. Game E, part 2. The first half of this game is lines 20–39 of Game D.

Second, we modify the game so as to "give up" (set *bad*) if the condition of line 62 is satisfied. In doing this, we may again decrease the probability that *bad* will be set to `true`. But the decrease is at most $1/2^\tau$ since, when the **else** clause of the new line 62 is executed (that is, $Y[m+1] \xleftarrow{\$} \{0,1\}^n$), T will equal T' with probability exactly $1/2^\tau$. Finally, we modify the game to give up (set *bad*) whenever $N \notin \{N_1, \ldots, N_q\}$, but $X[0] \doteq N \oplus L$ is already in Domain(π) when this is checked at line 50. The new game is called game E and it is shown in Figure 9. We note for future reference:

$$\Pr[\textit{bad} \text{ gets set in game D}] \leq \Pr[\textit{bad} \text{ gets set in game E}] + \frac{(c+2)(\sigma+2q+c+3)^2}{2^n} + \frac{1}{2^\tau}. \qquad (3)$$

Game F. We now examine game E and relate it to a final game, F. If *bad* is set to `true` in game E the reason is either that $X[0] = N \oplus L$ was found to be in the domain of π even though N is a new nonce, or else $X[c+1]$ was found to be in the domain of π when this was checked. In the latter case, how did $X[c+1]$ come to be in the domain of π? At least one of the following must be true:

—$X[c+1] = 0^n$. (The value 0^n was added to the domain of π at line 21.)
—For some $r \in [1..q]$, for some $j \in [0..m_r+1]$, $X[c+1] = X_r[j]$. (These values were added to the domain of π at line 39.)
—For some $i \in [0..c]$, $X[c+1] = X[i]$. (These values were added to the domain of π at lines 53, 57, and 61).

When *bad* is set to `true` we will assign responsibility for this event to exactly one index $r \in [1..q]$. We say that the *responsible index* is r where:

—If N is a new nonce and $X[0] \in$ Domain(π) at line 51, then the responsible index is the least $r \in [1..q]$ such that $X_r[j] = X[0]$ for some j. Otherwise,
—If $X[c+1] = 0^n$, then the responsible index is $r = 1$. Otherwise,

—If there is an $r \in [1..q]$ such that, for some $j \in [0..m_r + 1]$, $X[c+1] = X_r[j]$, then the responsible index is the least such value r. Otherwise,

—The responsible index is $r = 1$. (This last case can happen when $X[c+1] = X[i]$ for some $i \in [0..c]$.)

Partition the coins used in the running of game E into the coins s_0 used in the initialization step (line 21); the coins $s_1, \ldots, s_q$ used for processing message $M_1, \ldots, M_q$, respectively (line 31); and the coins s used to process the forgery attempt C (lines 52, 57, and 60). Suppose we eliminate the **for** statement at line 30, and execute lines 31–36 for some specific value of r. Call this game E_r. We make the crucial observation that if *bad* is set to `true` in game E using coins $(s_0, s_1, \ldots, s_q, s)$ then *bad* will *still* be set to `true` in game E_r using coins (s_0, s_r, s) when the responsible index is r. This follows from our definition of the responsible index. The only observation that is needed is that when $X[c+1] = X[i]$ for some $i \in [0..c]$, then, considering the least such i, if $X[i]$ was selected by assigning to it an already-selected $X_s[j]$-value, then the third case in the definition of the responsible index will result in the selection of an index r that forces *bad* to `true`.

By what we have said, one can bound the probability that *bad* gets set to `true` in game E by summing the probabilities that *bad* gets set to `true` in game E_r, where $r \in [1..q]$. Game E_r is precisely the game that was used to define the CMcoll_n; in particular, the probability that *bad* is set in E_r is $\mathrm{CMcoll}_n(c, m_r)$. We conclude that the probability that *bad* is set to `true` in game E_r is at most $\mathrm{CMcoll}_n(c, m_r)$. Thus the probability that *bad* gets set to `true` in game E is at most

$$\sum_{r=1}^{q} \mathrm{CMcoll}_n(c, m_r). \tag{4}$$

Summing Eqs. (1)–(4) gives that the adversary's chance of forgery is at most

$$\sum_{r \in [1..q]} \mathrm{Mcoll}_n(m_r) + \sum_{1 \le r < s \le q} \mathrm{MMcoll}_n(m_r, m_s) + \sum_{r=1}^{q} \mathrm{CMcoll}_n(c, m_r)$$
$$+ \frac{(\sigma + 2q + 1)^2 + 2(c+2)(\sigma + 2q + c + 3)}{2^{n+1}} + \frac{1}{2^\tau}.$$

Using that $(\sigma + \Delta)^2 - \sigma^2 \ge 2\sigma\Delta$ and $(\sigma + \Delta)^2 - \sigma^2 \ge \Delta^2$, we can increase σ by a small amount in order to compensate for the lower-order terms and clean up the expression. Namely, increasing σ by $2q + 1$ is enough to take care of the first addend, while increasing σ by $c + 2$ plus $2(c+2)$ plus $\sqrt{2}(c+3)$ is enough to take care of the second addend. So increasing σ by $2q + 5c + 11$ will take care of both. Letting $\bar{\sigma} = 2q + 5c + 11$, we thus have that the adversary's chance of forgery is at most

$$\sum_{1 \le r < s \le q} \mathrm{MMcoll}_n(q_r, q_s) + \sum_{r=1}^{q} \mathrm{CMcoll}_n(c, q_s) + \frac{\bar{\sigma}^2}{2} \cdot \frac{1}{2^n} + \frac{1}{2^\tau}.$$

This completes the proof of the structure lemma. □

8.3 Proof of the M- and MM-Collision Bounds (Lemma 2)

We assume that $m, \bar{m} < 2^{n-2}$, since the specified probability upper bound is meaningless (it exceeds 1) otherwise. According to remarks we have made earlier, this ensures that $\gamma_1, \ldots, \gamma_{\max\{m,\bar{m}\}}$, *huge* are distinct nonzero field elements.

We begin with the first inequality. There are $m + 3$ points in the set $\mathcal{X}_0$, and we claim that for any two of them, the probability that they coincide is at most $1/2^n$. This is enough to show the first inequality, that the probability of a collision within $\mathcal{X}_0$ is at most $\binom{m+3}{2} \cdot 2^{-n}$. There are a few cases to consider. Below, remember that L and R are random, and everything else is constant. The probabilities are over L, R. In the following, we let $i, i' \in [1..m-1]$, $i \neq i'$.

—$\Pr[X[-1] = X[0]] = \Pr[0^n = M[0] \oplus L] = 1/2^n$.
—$\Pr[X[-1] = X[i]] = \Pr[0^n = M[i] \oplus \gamma_i \cdot L \oplus R] = 1/2^n$.
—$\Pr[X[-1] = X[m]] = \Pr[0^n = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R] = 1/2^n$.
—$\Pr[X[-1] = X[m+1]] = \Pr[0^n = M[m+1] \oplus \gamma_m \cdot L \oplus R] = 1/2^n$.
—$\Pr[X[0] = X[i]] = \Pr[M[0] \oplus L = M[i] \oplus \gamma_i \cdot L \oplus R] = 1/2^n$.
—$\Pr[X[0] = X[m]] = \Pr[M[0] \oplus L = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R] = 1/2^n$.
—$\Pr[X[0] = X[m+1]] = \Pr[M[0] \oplus L = M[m+1] \oplus \gamma_m \cdot L \oplus R] = 1/2^n$.
—$\Pr[X[i] = X[i']] = \Pr[M[i] \oplus \gamma_i \cdot L = M[i'] \oplus \gamma_{i'} \cdot L] = \Pr[M[i] \oplus M[i'] = (\gamma_i \oplus \gamma_{i'}) \cdot L] = 1/2^n$ because $\gamma_i \neq \gamma_{i'}$.
—$\Pr[X[i] = X[m]] = \Pr[M[i] \oplus \gamma_i \cdot L \oplus R = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R] = \Pr[M[i] \oplus \gamma_i \cdot L = M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L] = \Pr[M[i] \oplus M[m] = (\gamma_m \oplus \mathit{huge} \oplus \gamma_i) \cdot L] = 1/2^n$ because $\gamma_i \oplus \gamma_m \neq \mathit{huge}$. The reason that $\gamma_i \oplus \gamma_m \neq \mathit{huge}$ is that *huge* begins with a 1 in bit position 1, while neither γ_i nor γ_m do, because $i, m \leq 2^{n-2}$ and $\gamma_i < 2i$, $\gamma_m \leq 2m$.
—$\Pr[X[i] = X[m+1]] = \Pr[M[i] \oplus \gamma_i \cdot L \oplus R = M[m+1] \oplus \gamma_m \cdot L \oplus R] = \Pr[M[i] \oplus M[m+1] = (\gamma_i \oplus \gamma_m) \cdot L] = 1/2^n$.
—$\Pr[X[m] = X[m+1]] = \Pr[M[m] \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R = M[m+1] \oplus \gamma_m \cdot L \oplus R] = \Pr[M[m] \oplus M[m+1] = \mathit{huge} \cdot L] = 1/2^n$.

This completes the first inequality.

For the second inequality, we wish to show that for any point in $\mathcal{X}$ and any point in $\bar{\mathcal{X}}$, the probability that they coincide is at most 2^{-n}. The result follows, since there are at most $(m+2)(\bar{m}+2)$ such pairs. Remember, below, that L, R, and $\bar{R}$ are random, and everything else is constant. We let $i \in [1..m-1]$ and $j \in [1..\bar{m}-1]$. As before, $\gamma_1, \ldots, \gamma_m$, *huge* are distinct nonzero points.

—$\Pr[X[0] = \bar{X}[0]] = \Pr[M[0] \oplus L = \bar{M}[0] \oplus L] = 0$, since $M[0] \neq \bar{M}[0]$ by assumption.
—$\Pr[X[0] = \bar{X}[j]] = \Pr[M[0] \oplus L = \bar{M}[j] \oplus \gamma_j \cdot L \oplus \bar{R}] = 1/2^n$ due to the influence of $\bar{R}$.
—$\Pr[X[0] = \bar{X}[\bar{m}] = \Pr[M[0] \oplus L = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus \bar{R}] = 1/2^n$ due to the influence of $\bar{R}$.
—$\Pr[X[0] = \bar{X}[\bar{m}+1] = \Pr[M[0] \oplus L = \bar{M}[\bar{m}+1] \oplus \gamma_{\bar{m}} \cdot L \oplus \bar{R}] = 1/2^n$ due to the influence of $\bar{R}$.

—$\Pr[X[i] = \bar{X}[j]] = \Pr[M[i] \oplus \gamma_i \cdot L \oplus R = \bar{M}[j] \oplus \gamma_j \cdot L \oplus \bar{R}] = 1/2^n$ due to the influence of $\bar{R}$.
—$\Pr[X[i] = \bar{X}[\bar{m}]] = \Pr[M[i] \oplus \gamma_i \cdot L \oplus R = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus huge) \cdot L \oplus \bar{R}] = 1/2^n$ due to the influence of $\bar{R}$.
—$\Pr[X[i] = \bar{X}[\bar{m}+1] = \Pr[M[i] \oplus \gamma_i \cdot L \oplus R = \bar{M}[\bar{m}+1] \oplus \gamma_{\bar{m}} \cdot L \oplus \bar{R}] = 1/2^n$ due to the influence of $\bar{R}$.
—$\Pr[X[m] = \bar{X}[\bar{m}] = 1/2^n$, as before, due to the influence of $\bar{R}$.
—$\Pr[X[m] = \bar{X}[\bar{m}+1] = 1/2^n$ for the same reason.
—$\Pr[X[m+1] = \bar{X}[\bar{m}+1] = 1/2^n$ for the same reason.

The remaining cases follow by symmetry. This completes the proof. □

8.4 Proof of the CM-Collision Bound (Lemma 3)

At the top level, we consider two cases: $N \neq \bar{N}$ and $N = \bar{N}$. The second of these will be analyzed by breaking into four subcases.

Case 1: $N \neq \bar{N}$. In this case there are two ways for *bad* to be set to `true`: it can happen at line 31 or line 44 in the game that defines the CMcoll_n collision probability (Figure 3). Let us first calculate the probability that *bad* is set to `true` at line 31, which is

$$\Pr[bad \text{ is set at line } 31] = \Pr[N \oplus L \in \{0^n, \bar{X}[1], \ldots, \bar{X}[\bar{m}+1]\}]$$

Note that the point $\bar{X}[0] = \bar{N} \oplus L$ in the domain of π has been omitted from set $B = \{0^n, \bar{X}[1], \ldots, \bar{X}[\bar{m}], \bar{X}[\bar{m}+1]\}$, and we know this point is different from $N \oplus L$ since $N \neq \bar{N}$. The probability above is taken over L and $\bar{R}$, where each $\bar{X}[i]$ implicitly depends on both. We claim that for each of the $\bar{m}+2$ values in B, the probability that $N \oplus L$ is equal to this particular value is exactly $1/2^n$. This is verified by

—$\Pr[N \oplus L = 0^n] = 1/2^n$ because of the random L.
—For any $j \in [1..\bar{m}-1]$, $\Pr[N \oplus L = \bar{X}[j]] = \Pr[N \oplus L = \bar{M}[j] \oplus \gamma_j \cdot L \oplus \bar{R}] = 1/2^n$ because of the random $\bar{R}$.
—Similarly, $\Pr[N \oplus L = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus huge) \cdot L \oplus \bar{R}] = 1/2^n$ because of the random $\bar{R}$.
—Similarly, $\Pr[N \oplus L = \bar{M}[\bar{m}+1]] = \Pr[N \oplus L = \text{Checksum}' \oplus \gamma_{\bar{m}} \cdot L \oplus \bar{R}] = 1/2^n$ because of the random $\bar{R}$.

We conclude that

$$\Pr[bad \text{ is set at line } 31] \leq \frac{\bar{m}+2}{2^n}. \tag{5}$$

We next show that

$$\Pr[X[c] \in \text{Domain}(\pi) \text{ at line } 40] \leq \frac{c+\bar{m}+3}{2^n}. \tag{6}$$

For this, let us define S to be

$$S = \{0^n, \bar{X}[0], \bar{X}[1], \ldots, \bar{X}[\bar{m}+1], X[0], X[1], \ldots, X[c-1]\}.$$

This is the domain of π at the time that line 40 is executed. The set has $c+\bar{m}+3$ points and we shall use the sum bound to see that the probability that $X[m]$ is one of these is at most $(c+\bar{m}+3)/2^n$. Namely,

—$\Pr[X[c] = 0^n] = \Pr[\text{len}(C[c]) \oplus (\gamma_m \oplus \mathit{huge}) \cdot L \oplus R = 0^n] = 1/2^n$ as the right-hand side of the equality sign does not depend on R.

—$\Pr[X[c] = \bar{X}[0]] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = \bar{N} \oplus L] = 1/2^n$ for the same reason.

—For $j \in [1..\bar{m}-1]$, $\Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = \bar{M}[j] \oplus \gamma_j \cdot L \oplus \bar{R}] = 1/2^n$ for the same reason.

—$\Pr[X[c] = \bar{X}[\bar{m}]] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = \bar{M}[\bar{m}] \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus \bar{R}] = 1/2^n$ for the same reason.

—$\Pr[X[c] = \bar{X}[\bar{m}+1]] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = \text{Checksum}' \oplus \gamma_{\bar{m}} \cdot L \oplus \bar{R}] = 1/2^n$ for the same reason.

—$\Pr[X[c] = X[0]] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = N \oplus L] = 1/2^n$ for the same reason.

—For $i \in [1..c-1]$, $X[i]$ is determined in one of two possible ways: either it is a value already placed into the Domain(π) (the **then** clause at line 37 was executed) or else it is a randomly selected value in $\{0,1\}^n$ (the **else** clause was executed). In the former case, the sum bound has already accounted for the probability of a collision with $X[i]$. In the latter case, the chance of the random value colliding with $X[c] = \text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R$ is $1/2^n$.

Equation (6) has now been established.

Next we observe that

$$\Pr[X[c+1] \in \text{Domain}(\pi) \text{ at line 44} \mid X[c] \notin \text{Domain}(\pi) \text{ at line 40}] \leq \frac{c+\bar{m}+4}{2^n}. \tag{7}$$

The reason is that, when the conditioning event happens, $Y[c]$ is selected as a random point in $\{0,1\}^n$ at line 40, which results in Checksum being a random value independent of the points in the domain of π, which results in $X[c+1]$ being a random value independent of the points in the domain of π. Since the domain of π has at most $1+(\bar{m}+2)+(c+1) = c+\bar{m}+4$ points at this time, Eq. (7) follows. Now, summing Eqs. (5)–(7) gives us that

$$\Pr[\mathit{bad} \text{ gets set} \mid \text{Case 1}] \leq \frac{3\bar{m}+2c+9}{2^n}. \tag{8}$$

Case 2A: $N = \bar{N}$ *and* $c \neq \bar{m}$. The next case we consider is when $N = \bar{N}$ and $c \neq \bar{m}$. Redefine S to be

$$S = \{0^n,\ \bar{X}[0], \ldots, \bar{X}[\bar{m}+1],\ X[1], \ldots, X[c-1]\}.$$

This is Domain(π) at the time line 40 is executed. We show that

$$\Pr[X[c] \in S \mid \text{Case 2a}] \leq \frac{c+\bar{m}+2}{2^n}. \tag{9}$$

To show this, one has as before to go through the $c + \bar{m} + 2$ points of S:

—$\Pr[X[c] = 0^n] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = 0^n] = 1/2^n$.
—$\Pr[X[c] = N \oplus L] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = N \oplus L] = 1/2^n$.
—For $j \in [1..\bar{m} - 1]$, $\Pr[X[c] = \bar{X}[j]] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = \bar{M}[j] \oplus \gamma_j \cdot L \oplus R] = \Pr[\text{len}(C[c]) \oplus \bar{M}[j] = (\gamma_j \oplus \gamma_c \oplus \mathit{huge}) \cdot L] = 1/2^n$ since $\gamma_j \oplus \gamma_c \neq \mathit{huge}$. The reason that $\gamma_j \oplus \gamma_c \neq \mathit{huge}$ is that $\gamma_j < 2j \leq 2\bar{m} \leq 2 \cdot 2^{n-2} = 2^{n-1}$, so γ_j begins with a 0-bit; and $\gamma_c < 2c \leq 2\bar{m} \leq 2 \cdot 2^{n-2} = 2^{n-1}$, so γ_c begins with a 0-bit; so the xor of γ_j and γ_c begins with a 0-bit, while *huge* begins with a 1-bit, so they are certainly unequal.
—$\Pr[X[c] = \bar{X}[\bar{m}]] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = \text{len}(\bar{M}[\bar{m}]) \oplus (\gamma_{\bar{m}} \oplus \mathit{huge}) \cdot L \oplus R] = \Pr[\text{len}(C[c]) \oplus \text{len}(\bar{M}[\bar{m}]) = (\gamma_c \oplus \gamma_{\bar{m}}) \cdot L] = 1/2^n$ since $\gamma_c \neq \gamma_{\bar{m}}$ (since $c \neq \bar{m}$).
—$\Pr[X[c] = \bar{X}[\bar{m} + 1]] = \Pr[\text{len}(C[c]) \oplus (\gamma_c \oplus \mathit{huge}) \cdot L \oplus R = \text{Checksum}' \oplus \gamma_{\bar{m}} \cdot L \oplus R] = \Pr[\text{len}(C[c]) \oplus \text{Checksum}' = (\gamma_c \oplus \mathit{huge} \oplus \gamma_{\bar{m}}) \cdot L] = 1/2^n$ as before.
—For $i \in [1..c - 1]$, either $X[i]$ was selected as a value already in $\text{Domain}(\pi)$, in which case the sum bound has already accounted for the probability of a collision with $X[c]$, or else $X[i]$ was selected as a new random value, in which case it has a $1/2^n$ chance of colliding with $X[c]$.

We have established (9). Next, as before, if $X[c] \notin S$ then $Y[c]$ is chosen at random, making Checksum random, and making $X[c + 1]$ random. Thus

$$\Pr[X[c+1] \in \text{Domain}(\pi) \text{ at line 44} \mid X[c] \notin \text{Domain}(\pi) \text{ at line 40}] \leq \frac{c + \bar{m} + 3}{2^n} \tag{10}$$

since the size of the domain of π at line 44 is at most $c + \bar{m} + 3$. Adding Eqs. (9) and (10) we have that

$$\Pr[\mathit{bad} \text{ gets set} \mid \text{Case 2A}] \leq \frac{2c + 2\bar{m} + 5}{2^n}. \tag{11}$$

Case 2B: $N = \bar{N}$ *and* $c = \bar{m}$ *and* $\exists\, a, a < c$, *such that* $C[a] \neq \bar{C}[a]$. In this case, let $a \geq 1$ be the smallest index such that $C[a] \neq \bar{C}[a]$. We claim that $Y[a]$ is almost certainly not in the range of π when this point is examined at line 37, when $i = a$. In fact, we claim something stronger: that $Y[a]$ is almost certainly different from every point in

$$S = \{L,\ \bar{Y}[0], \ldots, \bar{Y}[c+1],\ Y[1], \ldots, Y[a-1],\ Y[a+1], \ldots, Y[c-1]\}.$$

In particular,

$$\Pr[Y[a] \in S] \leq \frac{c + \bar{m}}{2^n}. \tag{12}$$

This is verified by going through each point in S, exactly as before. This time, for each point in S except $\bar{Y}[a]$, the probability that this point coincides with $Y[a]$ is exactly $1/2^n$. The probability that $\bar{Y}[a] = Y[a]$ is 0, since $C[a] \neq \bar{C}[a]$.

Now we modify the game that defines CMcoll_n so that $X[a]$ is always selected at random from $\{0,1\}^n$. If we bound the probability that *bad* gets set in this new game and then add to it the bound of Eq. (12), the result bounds the probability that *bad* gets set in Case 2B. From now on in this case analysis, assume this new game.

Next we claim that $X[c]$ is almost certainly different from $X[a]$:

$$\Pr[X[c] = X[a]] = \frac{1}{2^n}. \tag{13}$$

This is clear because, in the modified game we have described, $X[a]$ is now chosen at random, independent of $X[c] = \mathrm{len}(C[c]) \oplus (\mathit{huge} \oplus \gamma_c) \cdot L \oplus R$.

We may now modify the game once again so that $Y[c]$ is selected at random even in the case that $X[c] = X[a]$. Bounding the probability of *bad* being set in the new game, and adding in the bound of (13), serves to bound the probability of *bad* being set in the prior game.

Now we can look at the probability that $X[c+1] \in \mathrm{Domain}(\pi)$ when this is checked in the modified game. At this point, the domain of π contains the $2c+3$ points

$$\mathrm{Domain}^* = \{0^n,\ \bar{X}[0], \ldots, \bar{X}[\bar{m}+1],\ X[1], \ldots, X[a], \ldots, X[c]\}.$$

We want to know the probability that $X[c+1] = \mathrm{Checksum} \oplus \gamma_c \cdot L \oplus R$ is in this set. The value Checksum now contains the point $X[a]$, which, in the modified game, has just been selected at random and independent of all points in Domain^* with the exception of $X[a]$ itself. So for each of these $2c+2$ points, the probability that it coincides with $X[c+1]$ is $1/2^n$. For the one remaining point $X[a]$, note that $\Pr[X[c+1] = X[a]] = 1/2^n$, as the probability can be rewritten as $\Pr[\gamma_c \cdot L \oplus R = \mathrm{Checksum}']$, where Checksum′ (which is Checksum without the $X[a]$) is independent of L and R. Thus

$$\Pr[X[c+1] \in \mathrm{Domain}(\pi) \text{ in the modified game}] \le \frac{c+\bar{m}+3}{2^n}. \tag{14}$$

Summing Eqs. (12)–(14), we conclude that

$$\Pr[\mathit{bad} \text{ gets set} \mid \text{Case 2B}] \le \frac{2c+2\bar{m}+4}{2^n}. \tag{15}$$

Case 2C: $N = \bar{N}$ *and* $c = \bar{m}$ *and* $C[i] = \bar{C}[i]$ *for all* $1 \le i < c$ *and* $|C[c]| = |\bar{C}[c]|$. In this case, necessarily $C[c] \neq \bar{C}[c]$. Note that Checksum has a known value, which is different from Checksum′, being exactly $\mathrm{Checksum}' \oplus \bar{C}[c]\,0^* \oplus C[c]\,0^*$. The values $M[1], \ldots, M[c-1]$ are likewise known, being identical to $\bar{M}[1], \ldots, \bar{M}[c-1]$, respectively. We are interested in

$$\Pr[X[c+1] \in \{0^n, X[0], \ldots, X[c], \bar{X}[c+1]\}.$$

One goes through each of the points, as before, and sees that the probability that $X[c+1] = \mathrm{Checksum} \oplus \gamma_c \cdot L \oplus R$ is any one of them is $1/2^n$, except for the last point, for which the probability that they coincide is 0. Thus

$$\Pr[\mathit{bad} \text{ gets set} \mid \text{Case 2C}] \le \frac{c+2}{2^n}. \tag{16}$$

Case 2D: $N = \bar{N}$ *and* $c = \bar{m}$ *and* $C[i] = \bar{C}[i]$ *for all* $1 \le i < c$ *and* $|C[c]| \neq |\bar{C}[c]|$. For this case, we first claim that $X[c]$ is almost certainly not in the domain of π when this is inspected at line 40 of Figure 3. The method is as before. The point $X[c]$ is certain to be different from $\bar{X}[c]$, and its chance of coinciding with any of the $c+2$ points in $\{0^n, X[0], X[1], \ldots, X[c-1], \bar{X}[c+1]\}$ is easily verified to be $1/2^n$. Thus

$$\Pr[X[c] \in \text{Domain}(\pi) \text{ at line 40 }] \le \frac{c+2}{2^n}. \tag{17}$$

Proceeding as before,

$$\Pr[X[c+1] \in \text{Domain}(\pi) \text{ at line 44 } \mid X[c] \notin \text{Domain}(\pi) \text{ at line 38 }] \le \frac{c+3}{2^n} \tag{18}$$

since $c + 3$ bounds the size of the domain when line 44 is executed, and the conditioning event ensures a random value for $X[c + 1]$ that is independent of these points. Summing the bounds of Eqs. (17) and (18) gives

$$\Pr[X[c+1] \in \text{Domain}(\pi) \text{ at line 44 }] \le \frac{2c+5}{2^n}. \tag{19}$$

Conclusion. Taking the maximum from Eqs. (8), (11), (15), (16), and (19) we have

$$\Pr[\textit{bad} \text{ gets set }] \le \frac{3\bar{m} + 2c + 9}{2^n}$$

which is the lemma.

8.5 Proof of the Privacy Bound (Lemma 4)

The proof is straightforward compared to authenticity, so we quickly go though it. We begin by following the proof of the Structure Lemma (Section 8.2). Games A to D are defined as before, except that

—The second half of each game is omitted, since there is no forgery attempt in this context.
—Return the truncated final-ciphertext-blocks, instead of the full final-ciphertext blocks, as the games specify.

Focus on the (modified) game C, where we have now returned to the adversary A a random string of $|M_r| + \tau$ bits, whenever a query M_r is asked. Furthermore, the behavior of game C coincides with the behavior of the original game A unless the flag *bad* is set to `true`, at which point the two games diverge. Thus we can bound $\mathbf{Adv}^{\text{priv}}_{\text{OCB}[\text{Perm}(n),\tau]}(A)$ by bounding the probability that the flag *bad* is set to `true` in (the modified) game C, which is at most the probability that it gets set in Game D. From the same reasoning as in the structure lemma,

this is at most

$$\frac{(\sigma + 2q + 1)^2}{2^{n+1}} + \max_{\substack{m_1,\ldots,m_q \\ \sum m_i = \sigma \\ m_i \geq 1}} \left\{ \sum_{r \in [1..q]} \mathrm{Mcoll}_n(m_r) + \sum_{1 \leq r < s \leq q} \mathrm{MMcoll}_n(m_r, m_s) \right\}$$

which is precisely the bound given by the the lemma. □

APPENDIX: BRIEF HISTORY

An April '99 paper by Gligor and Donescu gives an authenticated-encryption scheme called PCBC [Gligor and Donescu 1999]. Though the mode is not correct, as pointed out by Jutla [2000a], it may have contributed to the subsequent development of correct modes. Jutla's paper [Jutla 2000a] gives the first correct schemes, IACBC and IAPM. Shortly after that paper appeared, Gligor and Donescu [2000a] described a different scheme, XCBC, which is similar to IACBC. The most conspicuous difference between XCBC and IACBC is the former's use of mod-2^n addition where the latter uses xor or mod-p addition, for p a prime just less than 2^n.

A first call by NIST for modes of operation brought contributions [Gligor and Donescu 2000b; Jutla 2000b] based on Gligor and Donescu [2000a]; Jutla [2000a], and a contribution by Rogaway [2000] that built on Jutla [2000a]. In Jutla [2000b], he employs a Gray-code ordering for combining basis offsets, a refinement independently introduced in Rogaway [2000].

A second call by NIST gave rise to Gligor and Donescu [2000c]; Jutla [2001b], and Rogaway et al. [20001a], which were revisions to Gligor and Donescu [2000b]; Jutla [2000b]; and Rogaway [2000], respectively. In Jutla [2001b], the author emphasized IAPM over IACBC, and he adopted "lazy mod-p addition" as described in Rogaway [2000]. Gligor and Donescu [2000c] described four authenticated-encryption modes, one of which, XECBS-XOR, is parallelizable. The modes adopt some features introduced in Rogaway [2000] to deal with messages of arbitrary length and to use a single block-cipher key. In Rogaway et al. [20001a], the authors settled on one mechanism to make offsets (three are described in Rogaway [2000]) and made further refinements to Rogaway [2000].

Briefly comparing OCB and IAPM, the latter uses two separate keys and is defined only for messages that are a multiple of the block length. Once a padding regime is included, say obligatory 10* padding, ciphertexts will be longer than OCB's by 1 to n bits. IAPM supports offset-production using either lazy mod-p addition or an xor-based scheme. The latter is not competitive with OCB in terms of key-setup costs.

An earlier version of this paper was published as Rogaway et al. [20001b].

The history above ignores associated patent applications.

ACKNOWLEDGMENTS

At CRYPTO '00, Virgil Gligor described Gligor and Donescu [2000a] to Rogaway; Charanjit Jutla gave a rump-session talk on Jutla [2000a]; and Elaine Barker announced a first modes-of-operation workshop organized by NIST. These

events inspired Rogaway [2000], which evolved into the current work. After the first workshop, NIST made a second call for proposals, and OCB took its final form in response to this call [Rogaway et al. 2001b]. We appreciate NIST's effort to solicit and evaluate modern modes of operation. Elaine Barker, Morris Dworkin, and Jim Foti are among those involved.

We received useful feedback from Michael Amling, Paulo Barreto, Johan Håstad, Hugo Krawczyk, Helger Lipmaa, David McGrew, Silvio Micali, Ilya Mironov, Alberto Pascual, Bart Preneel, Tom Shrimpton, and David Wagner. Special thanks to Michael and Ilya for their careful proofreading, and Helger for doing a state-of-the-art assembly implementation and providing associated timing data. Deepest thanks to Ted Krovetz who provided an implementation and performance figures. Thanks to the anonymous referees for their comments.

REFERENCES

AN, J. AND BELLARE, M. 2001. Does encryption with redundancy provide authenticity? In *Advances in Cryptology—EUROCRYPT 2001*. B. Pfitzmann, Ed. Lecture Notes in Computer Science, vol. 2045. Springer-Verlag, Berlin, 512–528. See also www-cse.ucsd.edu/users/mihir.

AOKI, K. AND LIPMAA, H. 2000. Fast implementations of AES candidates. In *The 3rd Advanced Encryption Standard Candidate Conference*. National Institute of Standards and Technology, New York, NY, USA, 106–120. See www.tml.hut.fi/~helger.

BELLARE, M., DESAI, A., JOKIPII, E., AND ROGAWAY, P. 1997. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97)*. ACM Press, New York, 394–403. See www.cs.ucdavis.edu/~rogaway.

BELLARE, M., DESAI, A., POINTCHEVAL, D., AND ROGAWAY, P. 1998. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology—CRYPTO '98*, H. Krawczyk, Ed. Lecture Notes in Computer Science, vol. 1462. Springer-Verlag, Berlin, 232–249. See www.cs.ucdavis.edu/~rogaway.

BELLARE, M., GUÉRIN, R., AND ROGAWAY, P. 1995. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *Advances in Cryptology—CRYPTO '95*, D. Coppersmith, Ed. Lecture Notes in Computer Science, vol. 963. Springer-Verlag, Berlin, 15–28. See www.cs.ucdavis.edu/~rogaway.

BELLARE, M., KILIAN, J., AND ROGAWAY, P. 2000. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences (JCSS) 61*, 3 (December), 362–399. Earlier version in CRYPTO '94. See www.cs.ucdavis.edu/~rogaway.

BELLARE, M. AND NAMPREMPRE, C. 2000. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology—ASIACRYPT '00*, T. Okamoto, Ed. Lecture Notes in Computer Science, vol. 1976. Springer-Verlag, Berlin, 531–545. See www-cse.ucsd.edu/users/mihir.

BELLARE, M. AND ROGAWAY, P. 2000. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient encryption. In *Advances in Cryptology—ASIACRYPT '00*, T. Okamoto, Ed. Lecture Notes in Computer Science, vol. 1976. Springer-Verlag, Berlin, 317–330. See www.cs.ucdavis.edu/~rogaway.

BLACK, J. AND URTUBIA, H. 2002. Side-channel attacks on symmetric encryption schemes: The case for authenticated encryption. In *Proceedings of the 11th USENIX Security Symposium*. USENIX, 327–338. See www.cs.colorado.edu/~jrblack/.

BLEICHENBACHER, D. 1998. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology—CRYPTO '98*. H. Krawczyk, Ed. Lecture Notes in Computer Science, vol. 1462. Springer-Verlag, Berlin, 1–12. See www.bell-labs.com/user/bleichen.

DOLEV, D., DWORK, C., AND NAOR, M. 2000. Non-malleable cryptography. *SIAM J. Comp. 3*, 2, 391–497. Earlier version appeared at STOC '91. See www.wisdom.weizmann.ac.il/~naor/.

GLIGOR, V. AND DONESCU, P. 1999. Integrity-aware PCBC encryption schemes. In *Security Protocols, 7th International Workshop*. B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, Eds. Lecture Notes in Computer Science, vol. 1796. Springer-Verlag, Berlin, 153–171.

GLIGOR, V. AND DONESCU, P. 2000a. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Manuscript, Aug. 18. See www.eng.umd.edu/~gligor.

GLIGOR, V. AND DONESCU, P. 2000b. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Contribution to NIST, Oct 27, 2000. See csrc.nist.gov/encryption/aes/modes.

GLIGOR, V. AND DONESCU, P. 2000c. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Contribution to NIST, Mar 30, 2001, rev. Apr 20, 2001. See csrc.nist.gov/encryption/aes/proposedmodes.

GLIGOR, V. AND DONESCU, P. 2002. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In *Fast Software Encryption, 8th International Workshop, FSE 2001*. M. Matsui, Ed. Lecture Notes in Computer Science, vol. 2355. Springer-Verlag, Berlin, 92–108. See www.ece.umd.edu/~gligor/.

GOLDWASSER, S. AND MICALI, S. 1984. Probabilistic encryption. *J. Comput. Syst. Sci. 28*, 270–299.

HALEVI, S. 2001. An observation regarding Jutla's modes of operation. Cryptology ePrint archive, reference number 2001/015, submitted Feb. 22, 2001, revised Apr. 2, 2001. See eprint.iacr.org.

JUTLA, C. 2000a. Encryption modes with almost free message integrity. Cryptology ePrint archive, reference number 2000/039, Aug. 1, 2000. See eprint.iacr.org.

JUTLA, C. 2000b. Encryption modes with almost free message integrity. Contribution to NIST. Undated manuscript, appearing Oct. 2000 at csrc.nist.gov/encryption/modes/workshop1.

JUTLA, C. 2001a. Encryption modes with almost free message integrity. In *Advances in Cryptology—EUROCRYPT 2001*. B. Pfitzmann, Ed. Lecture Notes in Computer Science, vol. 2045. Springer-Verlag, Berlin.

JUTLA, C. 2001b. Encryption modes with almost free message integrity. Contribution to NIST. Undated manuscript, posted May 24, 2001 at NIST web site csrc.nist.gov/encryption/modes/proposedmodes.

KATZ, J. AND YUNG, M. 2000a. Complete characterization of security notions for probabilistic private-key encryption. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000)*. ACM Press, New York, 245–254.

KATZ, J. AND YUNG, M. 2000b. Unforgeable encryption and adaptively secure modes of operation. In *Fast Software Encryption, 7th International Workshop, FSE 2000*. B. Schneier, Ed. Lecture Notes in Computer Science, vol. 1978. Springer-Verlag, Berlin, 284–299.

KRAWCZYK, H. 2001. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Advances in Cryptology—CRYPTO 2001*. J. Kilian, Ed. Lecture Notes in Computer Science, vol. 2139. Springer-Verlag, 310–331.

LIPMAA, H. 2001. Personal communications. July 2001. Further information available at www.tml.hut.fi/~helger.

LUBY, M. AND RACKOFF, C. 1988. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput. 17*, 2 (April), 373–386.

MANGER, J. 2001. A chosen ciphertext attack on RSA optimal asymmetric encryption padding OAEP as standardized in PKCS#1 v2.0. In *Advances in Cryptology—CRYPTO '01*. J. Kilian, Ed. Lecture Notes in Computer Science, vol. 2139. Springer-Verlag, Berlin, 230–238.

MEYER, C. H. AND MATYAS, S. M. 1982. *Cryptography: A New Dimension in Computer Data Security*. John Wiley and Sons, New York.

PRENEEL, B. 1998. Cryptographic primitives for information authentication—State of the art. In *State of the Art in Applied Cryptography, COSIC '97*. Lecture Notes in Computer Science, vol. 1528. Springer-Verlag, Berlin, 49–104.

ROGAWAY, P. 2000. OCB mode: Parallelizable authenticated encryption. Contribution to NIST, Oct. 16, 2000 (Preliminary version of the OCB algorithm). See csrc.nist.gov/encryption/modes/workshop1.

ROGAWAY, P. 2002. Authenticated-encryption with associated-data. In *ACM Conference on Computer and Communications Security (CCS-9)*. ACM Press, New York, 196–205.

ROGAWAY, P., BELLARE, M., BLACK, J., AND KROVETZ, T. 2001a. OCB mode. Contribution to NIST, Apr. 1, 2001, revised Apr. 18, 2001. See csrc.nist.gov/encryption/modes/proposedmodes.

ROGAWAY, P., BELLARE, M., BLACK, J., AND KROVETZ, T. 2001b. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM Conference on Computer and Communications Security (CCS-8)*. ACM Press, New York, 196–205.

RSA LABORATORIES. 1998. PKCS #1: RSA encryption standard, Version 1.5, PKCS #1: RSA cryptography specifications, Version 2.0, Sep. 1998, B. Kaliski and J. Staddon. See www.rsasecurity.com/rsalabs/pkcs/pkcs-1.

SCHROEPPEL, R. 2001. Personal communications.

STEINER, J., NEUMAN, C., AND SCHILLER, J. 1988. Kerberos: an authentication service for open network systems. In *Proceedings of the Winter 1988 Usenix Conference*. USENIX Association, 191–201.

US NATIONAL INSTITUTE OF STANDARDS. 2001. *Specification for the Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication 197. Based on J. Daemen and V. Rijmen, AES Proposal: Rijndael. Sep. 3, 1999. See www.nist.gov/aes.

VAUDENAY, S. 2002. Security flaws induced by CBC padding—applications to SSL, IPSEC, WTLS.... In *Advances in Cryptology—EUROCRYPT '02*. L. Knudsen, Ed. Lecture Notes in Computer Science, vol. 2332. Springer-Verlag, Berlin, 534–546. See /lasecwww.epfl.ch/~vaudenay/.

Received March 2002; revised April 2003; accepted March 2003

OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption *

Phillip Rogaway†
UC Davis & Chiang Mai Univ

Mihir Bellare‡
UC San Diego

John Black§
University of Nevada

Ted Krovetz¶
Digital Fountain

ABSTRACT

We describe a parallelizable block-cipher mode of operation that simultaneously provides privacy and authenticity. OCB encrypts-and-authenticates a nonempty string $M \in \{0,1\}^*$ using $\lceil |M|/n \rceil + 2$ block-cipher invocations, where n is the block length of the underlying block cipher. Additional overhead is small. OCB refines a scheme, IAPM, suggested by Charanjit Jutla. Desirable properties of OCB include: the ability to encrypt a bit string of arbitrary length into a ciphertext of minimal length; cheap offset calculations; cheap session setup; a single underlying cryptographic key; no extended-precision addition; a nearly optimal number of block-cipher calls; and no requirement for a random IV. We prove OCB secure, quantifying the adversary's ability to violate the mode's privacy or authenticity in terms of the quality of its block cipher as a pseudorandom permutation (PRP) or as a strong PRP, respectively.

Keywords

AES, authenticity, block ciphers, cryptography, encryption, integrity, modes of operation, provable security, standards

1. INTRODUCTION

BACKGROUND. An authenticated-encryption scheme is a shared-key encryption scheme whose goal is to provide *both* privacy *and* authenticity. The encryption algorithm takes a key, a plaintext, and a nonce (often called an IV), and it returns a ciphertext. The decryption algorithm takes a key, a ciphertext, and a nonce, and it returns either a plaintext or a special symbol, INVALID. In addition to the customary privacy goal, an authenticated-encryption scheme aims for authenticity: if an adversary should try to create some new ciphertext, the decryption algorithm will almost certainly regard it as INVALID.

An authenticated-encryption scheme can be constructed by appropriately combining an encryption scheme and a message authentication code (MAC), an approach used pervasively in practice and in standards. (Analyses of these methods are provided in [7, 24].) But an extremely attractive goal is an authenticated-encryption scheme having computational cost significantly lower than the cost to encrypt plus the cost to MAC. The classical approach for trying to do this is to encrypt-with-redundancy, where one appends a noncryptographic checksum to the message before encrypting it, typically with CBC mode. Many such schemes have been broken. Recently, however, Charanjit Jutla has proposed two authenticated-encryption schemes supported by a claim of provable security [21]. Virgil Gligor and Pompiliu Donescu have described a different authenticated-encryption scheme [15]. We continue in this line of work.

OCB MODE. This paper describes a new mode of operation, OCB, which refines one of Jutla's schemes, IAPM [21]. OCB (which stands for "offset codebook") retains the principal characteristics of IAPM: it is fully parallelizable and adds minor overhead compared to conventional, privacy-only modes. But OCB combines the following features:

- *Arbitrary-length messages + minimal-length ciphertexts*: Any string $M \in \{0,1\}^*$ can be encrypted; in particular, $|M|$ need not be a multiple of the block length n. What is more, the resulting ciphertexts are as short as possible; plaintexts are *not* padded to a multiple of n bits.

---

* A full version of this paper is available as [33].

† Dept. of Comp. Sci., Eng. II Bldg., Univ. of California, Davis, CA 95616 USA; and Dept. of Comp. Sci., Faculty of Science., Chiang Mai University, Chiang Mai 50200 Thailand. e-mail: rogaway@cs.ucdavis.edu web: www.cs.ucdavis.edu/~rogaway

‡ Dept. of Comp. Sci. & Eng., Univ. of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093 USA. e-mail: mihir@cs.ucsd.edu web: www-cse.ucsd.edu/users/mihir

§ Dept. of Comp. Sci., Univ. of Nevada, Reno, NV 89557 USA. e-mail: jrb@cs.unr.edu web: www.cs.unr.edu/~jrb

¶ Digital Fountain, 600 Alabama Street, San Francisco, CA 94110 USA. e-mail: tdk@acm.org

CCS'01, November 5±8, 2001, Philadelphia, Pennsylvania, USA.

- *Nearly optimal number of block-cipher calls*: OCB uses $\lceil |M|/n \rceil + 2$ block-cipher invocations (excluding a block-cipher call assumed to be made during session setup). (It is possible to make do with $\lceil |M|/n \rceil + 1$ calls, but such alternatives seem to be more complex or require a random IV.) Keeping low the number of block-cipher calls is especially important when messages are short.
- *Minimal requirements on nonces*: Like other encryption modes, OCB requires a nonce. The entity that encrypts chooses a new nonce for every message with the only restriction that no nonce is used twice. Schemes that require non-repeating nonces are less likely to be misused, and often more efficient, than those requiring random IVs.
- *Efficient offset calculations*: As with [15, 21], we require a sequence of *offsets*. We generate these in a particularly cheap way, each offset requiring just a few machine cycles. We avoid the use of extended-precision addition, which would introduce endian dependency and might make the scheme less attractive for dedicated hardware.
- *Single underlying key*: The key used for OCB is a single block-cipher key, and all block-cipher invocations are keyed by this one key, saving space and key-setup time.

Achieving the properties above has required putting together a variety of "tricks" that work together in just the right way. Many earlier versions of the algorithm were rejected by the authors because attacks were found or a proof could not be pushed through. We have found schemes of this sort to be amazingly "fragile"—tweak them a little and they break. We have concluded that, if the goals above are ever to be sought, they must be carefully addressed from the start.

PERFORMANCE. On a Pentium III processor, experiments by Lipmaa [25] show that OCB is about 6.5% slower than the privacy-only mode CBC. The cost of OCB is about 54% of the cost of CBC encryption combined with the CBC MAC. These figures assume a block cipher of AES128 [34].

In settings where there is adequate opportunity for parallelism, OCB will be faster than CBC. Parallelizability is important for obtaining the highest speeds from special-purpose hardware, and it may become useful on commodity processors. For special-purpose hardware, one may want to encrypt-and-authenticate at speeds near 10 Gbits/second—an impossible task, with today's technology, for modes like CBC encryption and the CBC MAC. (One could always create a mode that interleaves message blocks fed into separate CBC encryption or CBC MAC calculations, but that would be a new mode, and one with many drawbacks.) For commodity processors, there is an architectural trend towards highly pipelined machines with multiple instruction pipes and lots of registers. Optimally exploiting such features necessitates algorithms with plenty to do in parallel.

SECURITY PROPERTIES. We prove OCB secure, in the sense of reduction-based cryptography. Specifically, we prove indistinguishability under chosen-plaintext attack [3, 16] and authenticity of ciphertexts [7, 8, 22]. As shown in [7, 22], this combination implies indistinguishability under chosen-ciphertext attack (CCA) which, in turn, is equivalent to non-malleability [10] under CCA [4, 23]. (Non-malleability refers to an adversary's inability to modify a ciphertext in a way that makes related the two underlying plaintexts.) Our proof of privacy assumes that the underlying block cipher is good in the sense of a pseudorandom permutation (PRP) [6, 26], while our proof of authenticity assumes that the block cipher is a strong PRP [26]. Our results are quantitative; the security analysis is in the concrete-security paradigm.

We emphasize that OCB has stronger security properties than standard modes. In particular, non-malleability and indistinguishability under CCA are not achieved by CBC, or by any other standard mode, but these properties are achieved by OCB. We believe that the lack of strong security properties has been a problem for the standard modes of operation, because many users of encryption implicitly assume these properties when designing their protocols. For example, it is common to see protocols which use symmetric encryption in order to "bind together" the parts of a plaintext, or which encrypt related messages as a way to do a "handshake." Standard modes do not support such practices. This fact has sometimes led practitioners to incorrectly apply the standard modes, or to invent or select wrong ways to encrypt with authenticity (a well-known example is the use of PCBC mode [27] in Kerberos v.4 [29]). We believe that a mode like OCB is less likely to be misused because the usual abuses of privacy-only encryption become correct cryptographic techniques.

By way of comparison, a chosen-ciphertext attack by Bleichenbacher on the public-key encryption scheme of RSA PKCS #1, v.1, motivated the company that controls this de facto standard to promptly upgrade its scheme [9, 28]. In contrast, people seem to accept as a matter of course symmetric encryption schemes which are not even non-malleable. There would seem to be no technical reason to account for this difference in expectations.

THE FUTURE. We believe that *most* of the time privacy is desired, authenticity is too. As a consequence, fast authenticated encryption may quickly catch on. OCB has already appeared in one draft standard—the wireless LAN standard IEEE 802.11—and it is also under consideration by NIST.

2. PRELIMINARIES

NOTATION. If a and b are integers, $a \leq b$, then $[a..b]$ is the set $\{a, a+1, \ldots, b\}$. If $i \geq 1$ is an integer then $\mathsf{ntz}(i)$ is the number of trailing 0-bits in the binary representation of i (equivalently, $\mathsf{ntz}(i)$ is the largest integer z such that 2^z divides i). So, for example, $\mathsf{ntz}(7) = 0$ and $\mathsf{ntz}(8) = 3$.

A *string* is a finite sequence of symbols, each symbol being 0 or 1. The string of length 0 is called the *empty string* and is denoted ε. Let $\{0,1\}^*$ denote the set of all strings. If $A, B \in \{0,1\}^*$ then $A\,B$, or $A \parallel B$, is their concatenation. If $A \in \{0,1\}^*$ and $A \neq \varepsilon$ then $\mathsf{firstbit}(A)$ is the first bit of A and $\mathsf{lastbit}(A)$ is the last bit of A. Let i, n be nonnegative integers. Then 0^i and 1^i denote the strings of i 0's and 1's, respectively. Let $\{0,1\}^n$ denote the set of all strings of length n. If $A \in \{0,1\}^*$ then $|A|$ denotes the length of A, in bits, while $\|A\|_n = \max\{1, \lceil |A|/n \rceil\}$ denotes the length of A in n-bit blocks, where the empty string counts as one block. For $A \in \{0,1\}^*$ and $|A| \leq n$, $\mathsf{zpad}_n(A)$ is the string $A\,0^{n-|A|}$. With n understood we will write $A\,0^*$ for $\mathsf{zpad}_n(A)$. If $A \in \{0,1\}^*$ and $\tau \in [0..|A|]$ then $A[\text{first } \tau \text{ bits}]$ and $A[\text{last } \tau \text{ bits}]$ denote the first τ bits of A and the last τ bits of A, respectively. Both of these values are the empty string if $\tau = 0$. If $A, B \in \{0,1\}^*$ then $A \oplus B$ is the bitwise xor of $A[\text{first } \ell \text{ bits}]$ and $B[\text{first } \ell \text{ bits}]$, where $\ell = \min\{|A|, |B|\}$ (where $\varepsilon \oplus A = A \oplus \varepsilon = \varepsilon$). So, for example, $1001 \oplus 11 = 01$. If $A = a_{n-1} \cdots a_1 a_0 \in \{0,1\}^n$ then $\mathsf{str2num}(A)$ is the number $\sum_{i=0}^{n-1} 2^i a_i$. If $a \in [0..2^n - 1]$ then $\mathsf{num2str}_n(a)$ is the n-

bit string A such that $\mathsf{str2num}(A) = a$. Let $\mathsf{len}_n(A) = \mathsf{num2str}_n(|A|)$. We omit the subscript when n is understood.

If $A = a_{n-1}a_{n-2}\cdots a_1a_0 \in \{0,1\}^n$ then $A{\ll}1$ is the n-bit string $a_{n-2}a_{n-3}\cdots a_1a_00$ which is a left shift of A by one bit (the first bit of A disappearing and a zero coming into the last bit), while $A{\gg}1$ is the n-bit string $0a_{n-1}a_{n-2}\ldots a_2a_1$ which is a right shift of A by one bit (the last bit disappearing and a zero coming into the first bit).

In pseudocode we write "Partition M into $M[1]\cdots M[m]$" as shorthand for "Let $m = \|M\|_n$ and let $M[1],\ldots,M[m]$ be strings such that $M[1]\cdots M[m] = M$ and $|M[i]| = n$ for $1 \le i < m$." We write "Partition $\mathcal{C}$ into $C[1]\cdots C[m]\ T$" as shorthand for "if $|\mathcal{C}| < \tau$ then return INVALID. Otherwise, let $C = \mathcal{C}$[first $|\mathcal{C}| - \tau$ bits], let $T = \mathcal{C}$[last τ bits], let $m = \|C\|_n$, and let $C[1],\ldots,C[m]$ be strings such that $C[1]\cdots C[m] = C$ and $|C[i]| = n$ for $1 \le i < m$. Recall that $\|M\|_n = \max\{1, \lceil |M|/n \rceil\}$, so the empty string partitions into $m = 1$ block, that one block being the empty string.

THE FIELD WITH 2^n POINTS. Let $\mathrm{GF}(2^n)$ denote the field with 2^n points. We interchangeably think of a point a in $\mathrm{GF}(2^n)$ in any of the following ways: (1) as an abstract point in a field; (2) as an n-bit string $a_{n-1}\ldots a_1a_0 \in \{0,1\}^n$; (3) as a formal polynomial $a(\mathrm{x}) = a_{n-1}\mathrm{x}^{n-1} + \cdots + a_1\mathrm{x} + a_0$ with binary coefficients; (4) as an integer between 0 and $2^n - 1$, where the string $a \in \{0,1\}^n$ corresponds to the number $\mathsf{str2num}(a)$. For example, one can regard the string $a = 0^{125}101$ as a 128-bit string, as the number 5, as the polynomial $\mathrm{x}^2 + 1$, or as an abstract point in $\mathrm{GF}(2^{128})$.

To add two points in $\mathrm{GF}(2^n)$, take their bitwise xor. We denote this operation by $a \oplus b$. To multiply two points in the field, first fix an irreducible polynomial $p_n(\mathrm{x})$ having binary coefficients and degree n: say the lexicographically first polynomial among the irreducible degree n polynomials having a minimum number of nonzero coefficients. For $n = 128$, the indicated polynomial is $p_{128}(\mathrm{x}) = \mathrm{x}^{128}+\mathrm{x}^7+\mathrm{x}^2+\mathrm{x}+1$. To multiply $a, b \in \mathrm{GF}(2^n)$, which we denote $a \cdot b$, regard a and b as polynomials $a(\mathrm{x}) = a_{n-1}\mathrm{x}^{n-1} + \cdots + a_1\mathrm{x} + a_0$ and $b(\mathrm{x}) = b_{n-1}\mathrm{x}^{n-1} + \cdots + b_1\mathrm{x} + b_0$, form their product $c(\mathrm{x})$ over $\mathrm{GF}(2)$, and take the remainder one gets when dividing $c(\mathrm{x})$ by $p_n(\mathrm{x})$.

It is computationally simple to multiply $a \in \{0,1\}^n$ by x. We illustrate the method for $n = 128$, in which case multiplying $a = a_{n-1}\cdots a_1a_0$ by x yields $a_{n-1}\mathrm{x}^n + a_{n-2}\mathrm{x}^{n-1} + a_1\mathrm{x}^2 + a_0\mathrm{x}$. Thus, if the first bit of a is 0, then $a \cdot \mathrm{x} = a{\ll}1$. If the first bit of a is 1 then we must add x^{128} to $a{\ll}1$. Since $p_{128}(\mathrm{x}) = \mathrm{x}^{128} + \mathrm{x}^7 + \mathrm{x}^2 + \mathrm{x} + 1 = 0$ we know that $\mathrm{x}^{128} = \mathrm{x}^7 + \mathrm{x}^2 + \mathrm{x} + 1$, so adding x^{128} means to xor by $0^{120}10000111$. In summary, when $n = 128$,

$$a \cdot \mathrm{x} = \begin{cases} a{\ll}1 & \text{if } \mathsf{firstbit}(a) = 0 \\ (a{\ll}1) \oplus 0^{120}10000111 & \text{if } \mathsf{firstbit}(a) = 1 \end{cases}$$

It is similarly easy to divide $a \in \{0,1\}^{128}$ by x (i.e., to multiply a by the multiplicative inverse of x). If the last bit of a is 0, then $a \cdot \mathrm{x}^{-1}$ is $a{\gg}1$. If the last bit of a is 1 then we must add (xor) to $a{\gg}1$ the value x^{-1}. Since $\mathrm{x}^{128} = \mathrm{x}^7+\mathrm{x}^2+\mathrm{x}+1$ we have that $\mathrm{x}^{-1} = \mathrm{x}^{127} + \mathrm{x}^6 + \mathrm{x} + 1 = 10^{120}1000011$. In summary, when $n = 128$,

$$a \cdot \mathrm{x}^{-1} = \begin{cases} a{\gg}1 & \text{if } \mathsf{lastbit}(a) = 0 \\ (a{\gg}1) \oplus 10^{120}1000011 & \text{if } \mathsf{lastbit}(a) = 1 \end{cases}$$

If $L \in \{0,1\}^n$ and $i \ge -1$, we write $L(i)$ as shorthand for $L \cdot \mathrm{x}^i$. Using the equations just given, we have an easy way to compute from L the values $L(-1)$, $L(0)$, $L(1)$,..., $L(\mu)$, where μ is small number.

GRAY CODES. For $\ell \ge 1$, a Gray code is an ordering $\gamma^\ell = (\gamma_0^\ell\ \ \gamma_1^\ell\ \ \ldots\ \ \gamma_{2^\ell-1}^\ell)$ of $\{0,1\}^\ell$ such that successive points differ (in the Hamming sense) by just one bit. For n a fixed number, OCB makes use of the "canonical" Gray code $\gamma = \gamma^n$ constructed by $\gamma^1 = (0\ 1)$ and, for $\ell > 0$, $\gamma^{\ell+1} = (0\gamma_0^\ell\ 0\gamma_1^\ell\ \cdots\ 0\gamma_{2^\ell-2}^\ell\ 0\gamma_{2^\ell-1}^\ell\ 1\gamma_{2^\ell-1}^\ell\ 1\gamma_{2^\ell-2}^\ell\ \cdots\ 1\gamma_1^\ell\ 1\gamma_0^\ell)$. It is easy to see that γ is a Gray code. What is more, for $1 \le i \le 2^n - 1$, $\gamma_i = \gamma_{i-1} \oplus (0^{n-1}1{\ll}\mathsf{ntz}(i))$. This makes it easy to compute successive points.

We emphasize the following characteristics of the Gray-code values $\gamma_1, \gamma_2, \ldots, \gamma_{2^n-1}$: that they are distinct and different from 0; that $\gamma_1 = 1$; and that $\gamma_i < 2i$.

Let $L \in \{0,1\}^n$ and consider the problem of successively forming the strings $\gamma_1 \cdot L$, $\gamma_2 \cdot L$, $\gamma_3 \cdot L$, ..., $\gamma_m \cdot L$. Of course $\gamma_1 \cdot L = 1 \cdot L = L$. Now, for $i \ge 2$, assume one has already produced $\gamma_{i-1} \cdot L$. Since $\gamma_i = \gamma_{i-1} \oplus (0^{n-1}1{\ll}\mathsf{ntz}(i))$ we know that $\gamma_i \cdot L = (\gamma_{i-1} \oplus (0^{n-1}1{\ll}\mathsf{ntz}(i))) \cdot L = (\gamma_{i-1} \cdot L) \oplus (0^{n-1}1{\ll}\mathsf{ntz}(i)) \cdot L = (\gamma_{i-1} \cdot L) \oplus (L \cdot \mathrm{x}^{\mathsf{ntz}(i)}) = (\gamma_{i-1} \cdot L) \oplus L(\mathsf{ntz}(i))$. That is, the ith word in the sequence $\gamma_1 \cdot L, \gamma_2 \cdot L, \gamma_3 \cdot L, \ldots$ is obtained by xoring the previous word with $L(\mathsf{ntz}(i))$. Had the sequence we were considering been $\gamma_1 \cdot L \oplus R$, $\gamma_2 \cdot L \oplus R$, $\gamma_3 \cdot L \oplus R$, ..., the ith word would be formed in the same way for $i \ge 2$, but the first word in the sequence would have been $L \oplus R$ instead of L.

3. THE SCHEME

PARAMETERS. To use OCB one must specify a block cipher and a tag length. The *block cipher* is a function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, for some number n, where each $E(K,\cdot) = E_K(\cdot)$ is a permutation on $\{0,1\}^n$. Here $\mathcal{K}$ is the set of possible keys and n is the block length. Both are arbitrary, though we insist that $n \ge 64$, and we discourage $n < 128$. The *tag length* is an integer $\tau \in [0..n]$. By trivial means, the adversary will be able to forge a valid ciphertext with probability $2^{-\tau}$. The popular block cipher to use with OCB is likely to be AES [34]. As for the tag length, a suggested default of $\tau = 64$ is reasonable. Tags of 32 bits are standard in retail banking. Tags of 96 bits are used in IPSec.

We let OCB-E denote the OCB mode of operation using block cipher E and an unspecified tag length. We let $\mathrm{OCB}[E,\tau]$ denote the OCB mode of operation using block cipher E and tag length τ.

NONCES. Encryption under OCB mode requires an n-bit nonce, N. The nonce would typically be a counter (maintained by the sender) or a random value (selected by the sender). Security is maintained even if the adversary can control the nonce, subject to the constraint that no nonce may be repeated within the current session (that is, during the period of use of the current encryption key). The nonce need not be random, unpredictable, or secret.

The nonce N is needed both to encrypt and to decrypt. Typically it would be communicated, in the clear, along with the ciphertext. However, it is out-of-scope how the nonce is communicated to the party who will decrypt. In particular, we do not regard the nonce as part of the ciphertext.

DEFINITION OF THE MODE. See Figure 1 for an illustration of OCB, and see Figure 2 for the definition. The latter figure defines OCB encryption and decryption, while the key space $\mathcal{K}$ is the key space for the underlying block cipher E.

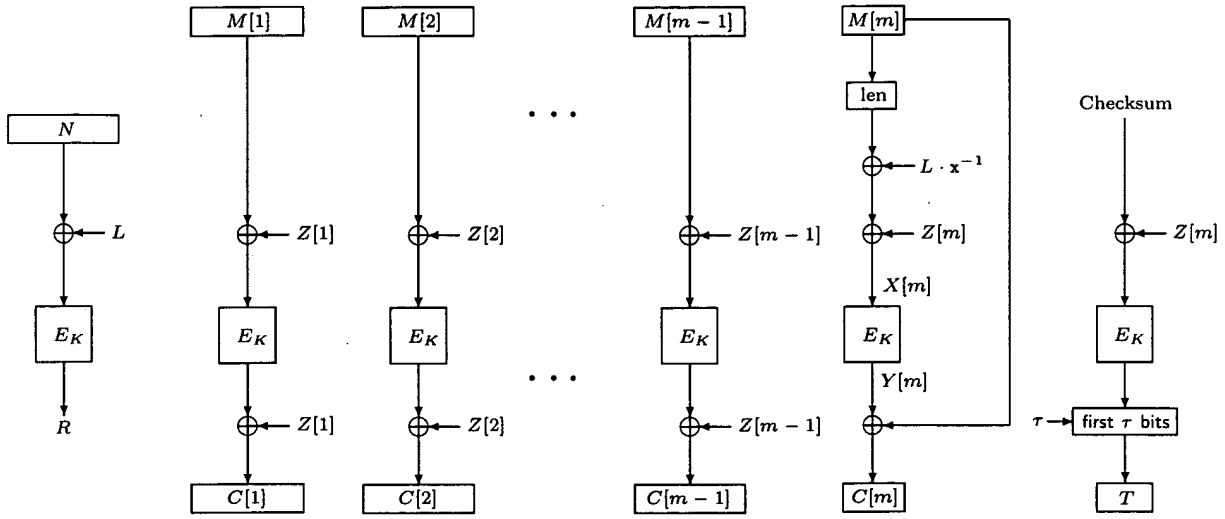


Figure 1: Illustration of OCB encryption. The message is $M = M[1]M[2]\cdots M[m-1]M[m]$ and the nonce is N. The resulting ciphertext is $\mathcal{C} = C[1]C[2]C[3]\cdots C[m-1]C[m]\,T$. The Checksum is $M[1]\oplus\cdots\oplus M[m-1]\oplus C[m]\,0^*\oplus Y[m]$. Define $L = E_K(0^n)$, $Z[1] = L \oplus R$, and, for $i \geq 2$, $Z[i] = Z[i-1] \oplus L(\mathsf{ntz}(i))$.

Algorithm $\mathrm{OCB.Enc}_K\ (N, M)$
Partition M into $M[1]\cdots M[m]$
$L \leftarrow E_K(0^n)$
$R \leftarrow E_K(N \oplus L)$
for $i \leftarrow 1$ **to** m **do** $Z[i] = \gamma_i \cdot L \oplus R$
for $i \leftarrow 1$ **to** $m-1$ **do** $C[i] \leftarrow E_K(M[i] \oplus Z[i]) \ \oplus\ Z[i]$
$X[m] \leftarrow \mathrm{len}(M[m]) \oplus L \cdot \mathrm{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$
$C[m] \leftarrow Y[m] \oplus M[m]$
$C \leftarrow C[1]\cdots C[m]$
$\mathrm{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
$T \leftarrow E_K(\mathrm{Checksum} \oplus Z[m])$ [first τ bits]
return $\mathcal{C} \leftarrow C \,\|\, T$

Algorithm $\mathrm{OCB.Dec}_K\ (N, \mathcal{C})$
Partition $\mathcal{C}$ into $C[1]\cdots C[m]\,T$
$L \leftarrow E_K(0^n)$
$R \leftarrow E_K(N \oplus L)$
for $i \leftarrow 1$ **to** m **do** $Z[i] = \gamma_i \cdot L \oplus R$
for $i \leftarrow 1$ **to** $m-1$ **do** $M[i] \leftarrow E_K^{-1}(C[i] \oplus Z[i]) \ \oplus\ Z[i]$
$X[m] \leftarrow \mathrm{len}(C[m]) \oplus L \cdot \mathrm{x}^{-1} \oplus Z[m]$
$Y[m] \leftarrow E_K(X[m])$
$M[m] \leftarrow Y[m] \oplus C[m]$
$M \leftarrow M[1]\cdots M[m]$
$\mathrm{Checksum} \leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]\,0^* \oplus Y[m]$
$T' \leftarrow E_K(\mathrm{Checksum} \oplus Z[m])$ [first τ bits]
if $T = T'$ **then return** M
else return INVALID

Figure 2: Definition of OCB. Encryption and decryption are specified, while key generation chooses a random element from the key space $\mathcal{K}$ of the block cipher.

AN EQUIVALENT DESCRIPTION. The following description may clarify what a typical implementation might do.

Key generation: Choose a random key $K \xleftarrow{R} \mathcal{K}$ for the block cipher. The key K is provided to both the entity that encrypts and the entity that decrypts.

Session setup: For the party that encrypts, do any key-setup associated to block-cipher enciphering. For the party that decrypts, do any key-setup associated to block-cipher enciphering and deciphering. Let $L \leftarrow E_K(0^n)$. Let m bound the maximum number of n-bit blocks that any message which will be encrypted or decrypted may have. Let $\mu \leftarrow \lceil \log_2 m \rceil$. Let $L(0) \leftarrow L$ and, for $i \in [1..\mu]$, compute $L(i) \leftarrow L(i-1) \cdot \mathrm{x}$ using a shift and a conditional xor, as described in Section 2. Compute $L(-1) \leftarrow L \cdot \mathrm{x}^{-1}$ using a shift and a conditional xor, as described in Section 2. Save the values $L(-1)$, $L(0)$, $L(1)$, ..., $L(\mu)$ in a table.

Encryption: To encrypt plaintext $M \in \{0,1\}^*$ using key K and nonce $N \in \{0,1\}^n$, obtaining a ciphertext $\mathcal{C}$, do the following. Let $m \leftarrow \lceil |M|/n \rceil$. If $m = 0$ then let $m \leftarrow 1$. Let $M[1], \ldots, M[m]$ be strings such that $M[1]\cdots M[m] = M$ and $|M[i]| = n$ for $i \in [1..m-1]$. Let $\mathrm{Offset} \leftarrow E_K(N \oplus L)$. Let $\mathrm{Checksum} \leftarrow 0^n$. For $i \leftarrow 1$ to $m-1$, do the following: let $\mathrm{Checksum} \leftarrow \mathrm{Checksum} \oplus M[i]$; let $\mathrm{Offset} \leftarrow \mathrm{Offset} \oplus L(\mathsf{ntz}(i))$; let $C[i] \leftarrow E_K(M[i] \oplus \mathrm{Offset}) \oplus \mathrm{Offset}$. Let $\mathrm{Offset} \leftarrow \mathrm{Offset} \oplus L(\mathsf{ntz}(m))$. Let $Y[m] \leftarrow E_K(\mathrm{len}(M[m]) \oplus L(-1) \oplus \mathrm{Offset})$. Let $C[m] \leftarrow M[m]$ xored with the first $|M[m]|$ bits of $Y[m]$. Let $\mathrm{Checksum} \leftarrow \mathrm{Checksum} \oplus Y[m] \oplus C[m]\,0^*$. Let T be the first τ bits of $E_K(\mathrm{Checksum} \oplus \mathrm{Offset})$. The ciphertext is $\mathcal{C} = C[1]\cdots C[m-1]C[m]\,T$. It must be communicated along with the nonce N.

Decryption: To decrypt ciphertext $\mathcal{C} \in \{0,1\}^*$ using key K and nonce $N \in \{0,1\}^n$, obtaining a plaintext $M \in \{0,1\}^*$ or an indication INVALID, do the following. If $|\mathcal{C}| < \tau$ then return INVALID (the ciphertext has been rejected). Otherwise let C be the first $|\mathcal{C}| - \tau$ bits of $\mathcal{C}$ and let T be the remaining τ bits. Let $m \leftarrow \lceil |C|/n \rceil$. If $m = 0$ then let $m = 1$. Let $C[1], \ldots, C[m]$ be strings such that $C[1]\cdots C[m] = C$ and $|C[i]| = n$ for $i \in [1..m-1]$. Let $\mathrm{Offset} \leftarrow E_K(N \oplus L)$. Let $\mathrm{Checksum} \leftarrow 0^n$. For $i \leftarrow 1$ to $m-1$, do the following: let

Offset $\leftarrow$ Offset$\oplus L(\mathsf{ntz}(i))$; let $M[i] \leftarrow E_K^{-1}(C[i] \oplus \text{Offset}) \oplus$ Offset; let Checksum $\leftarrow$ Checksum $\oplus\, M[i]$. Let Offset $\leftarrow$ Offset $\oplus\, L(\mathsf{ntz}(m))$. Let $Y[m] \leftarrow E_K(\mathsf{len}(C[m]) \oplus L(-1) \oplus$ Offset). Let $M[m] \leftarrow C[m]$ xored with the first $|C[m]|$ bits of $Y[m]$. Let Checksum $\leftarrow$ Checksum$\oplus Y[m] \oplus C[m]\,0^*$. Let T' be the first τ bits of E_K(Checksum$\oplus$Offset). If $T \neq T'$ then return INVALID (the ciphertext has been rejected). Otherwise, the plaintext is $M = M[1] \cdots M[m-1]M[m]$.

4. DISCUSSION

OCB has been designed to have a variety of desirable properties. Some of these have been discussed in the Introduction. We extend that discussion here.

ARBITRARY-LENGTH MESSAGES AND NO CIPHERTEXT EXPANSION. One of the key characteristics of OCB is that any string $M \in \{0,1\}^*$ can be encrypted, and doing this yields a ciphertext $\mathcal{C}$ of length $|M| + \tau$. That is, the length of the "ciphertext core"—the portion $C = C[1] \cdots C[m]$ of the ciphertext that excludes the tag—is the same as the length of the message M. This is better, by up to n bits, than what one gets with conventional padding.

SINGLE BLOCK-CIPHER KEY. OCB makes use of just one block-cipher key, K. While $L = E_K(0^n)$ functions rather like a key and would normally be computed at session-setup time, and while standard key-separation techniques can always be used to obtain many keys from one, the point is that, in OCB, all block-cipher invocations use the one key K. Thus only one block-cipher key needs to be setup, saving on storage space and key-setup time.

WEAK NONCE REQUIREMENTS. We believe that modes of operation that require a random IV are often misused. As an example, consider CBC mode, where $C[i] = E_K(M[i] \oplus C[i-1])$ and $C[0] = \text{IV}$. Many standards and many books (e.g., Schneier, *Applied Cryptography*, 2nd edition, p. 194]) suggest that the IV may be a fixed value, a counter, a timestamp, or the last block of ciphertext from the previous message. But if it is any of these things one certainly will not achieve any of the standard definitions of security [3, 16].

It is sometimes suggested that a mode which needs a random IV is preferable to one that needs a nonce: it is said that *state* is needed for a nonce, but not for making random bits. We find this argument wrong. First, a random value of sufficent length can always be used as a nonce, but a nonce can not be used as a random value. Second, the manner in which systems provide "random" IVs is invariably stateful anyway: unpredictable bits are too expensive to harvest for each IV, so one does this rarely, using state to generate pseudorandom bits from unpredictable bits harvested before. Third, the way to generate pseudorandom bits needs to use cryptography, so the prevalence of non-cryptographic pseudorandom number generators routinely results in implementation errors. Next, nonce-based schemes facillitate replay-detection with constant space and no added cryptography. Finally, nonces can be communicated using fewer bits, without additional cryptography.

ON-LINE. OCB encryption and decryption are "on line" in the sense that one does not need to know the length of the message in advance of encrypting or decrypting it. Instead, messages can be processed as one goes along, using constant memory, continuing until there is an indication that the message is over.

ENDIAN NEUTRALITY. In contrast to a scheme based on mod-p arithmetic (for p a prime just less than 2^n) or mod-2^n arithmetic, there is almost no endian-favoritism implicit in the definition of OCB. (The exception is that, because of our use of standard mathematical conventions, the left shift used for forming $L(i+1)$ from $L(i)$ is more convenient under a big-endian convention, as is the right shift used for forming $L(-1) = L \cdot \mathrm{x}^{-1}$ from L.)

OPTIONAL PRE-PROCESSING. Implementations can choose how many $L(i)$ values to precompute. As only one block-cipher call is needed to compuet these values from K, plus some shifts and conditional xors, it is feasible to do no preprocessing: OCB-AES is appropriate even when each session is a single, short message.

PROVABLE SECURITY. Provable security has become a popular goal for practical protocols. This is because it provides the best way to gain assurance that a cryptographic scheme does what it is should. For a scheme which enjoys provable security one does not need to consider attacks on the scheme, since successful ones imply successful attacks on some simpler object.

When we say that "OCB is provably secure" we are asserting the existence of two theorems. One says that if an adversary A could do a good job at forging ciphertexts with OCB$[E,\tau]$ (the adversary does this much more than a $2^{-\tau}$ fraction of the time) then there would be an adversary B that does a good job at distinguishing $(E_K(\cdot), E_K^{-1}(\cdot))$, for a random key K, from $(\pi(\cdot), \pi^{-1}(\cdot))$, for a random permutation $\pi \in \text{Perm}(n)$. The other theorem says that if an adversary A could do a good job at distinguishing OCB$[E,\tau]$-encrypted messages from random strings, then there would be an adversary B that does a good job at distinguishing $E_K(\cdot)$, for a random key K, from $\pi(\cdot)$, for a random permutation $\pi \in \text{Perm}(n)$. Theorems of this sort are called *reductions*. In cryptography, provable security means giving reductions (along with the associated definitions).

Provable security begins with Goldwasser and Micali [16], though the style of provable security which we use here—where the primitive is a block cipher, the scheme is a usage mode, and the analysis is concrete (no asymptotics)—is the approach of Bellare and Rogaway [3, 5, 6].

It is not enough to know that there is a provable-security result; one should also understand the definitions and the bounds. We have already sketched the definitions. When we speak of the bounds we are addressing "how effective is the adversary B in terms of the efficacy of adversary A" (where A and B are as above). For OCB, the bounds can be roughly summarized as follows. An adversary can always forge with probability $1/2^\tau$. Beyond this, the maximal added advantage is at most $\sigma^2/2^n$, where σ is the total number of blocks the adversary sees. The privacy bound likewise degrades as $\sigma^2/2^n$. The conclusion is that one is safe using OCB as long as the underlying block cipher is secure and σ is small compared to $2^{n/2}$. This is the same security degradation one observes for CBC encryption and in the bound for the CBC MAC [3, 6].

COMPARISON WITH JUTLA'S BOUND. More precisely, but still ignoring lower-order terms, our privacy and authenticity bounds are $1.5\,\sigma^2/2^n$, while Jutla's authenticity bound is insignificantly worse at $2\,\sigma^2/2^n$ and his privacy bound, rescaled to $[0,1]$, looks insignificantly worse at $3\,\sigma^2/2^n$ [20]. Magnifying the latter difference is that the privacy re-

sults assume different defintions. Jutla adopts the find-then-guess definition of privacy [3, 16], while we use an indistinguishability-from-random-bits definition. The former captures an adversary's inability to distinguish ciphertexts for a pair adversarilly-selected, equal-length plaintexts. The latter captures an adversary's inability to distinguish a ciphertext from a random string of the same length. Indistinguishability-from-random-bits implies find-then-guess security, and by a tight reduction, but find-then-guess secure does not imply indistinguishability-from-random-bits. Still, Jutla's scheme probably satisfies the stronger definition.

SIMPLICITY. Simplicity has been a central design goal. Some of OCB's characteristics that contribute to simplicity are:

- Short and full final-message-blocks are handled without making a special case: the treatment of all messages is uniform, regardless of their length.
- Only the simplest form of padding is used: append a minimal number of 0-bits to make a string whose length is a multiple of n. This method is computationally fastest and helps avoid a proliferation of cases in the analysis.
- Only one algebraic structure is used throughout the algorithm: the finite field GF(2^n).
- In forming the sequence of offsets, Gray-code coefficients are taken monotonically, starting at 1 and stopping at m. One never goes back to some earlier offset, uses a peculiar starting point, or forms more offsets than there are blocks.

NOT FIXING HOW THE NONCE IS COMMUNICATED. We do not specify how the nonce is chosen or communicated. Formally, it is not part of the ciphertext (though the receiving party needs it to decrypt). In many contexts, there is already a natural value to use as a nonce (e.g., a sequence number already present in a protocol flow, or implicit because the parties are communicating over a reliable channel). Even when a protocol is designed from scratch, the number of bits needed to communicate the nonce will vary.

NOT FIXING THE TAG LENGTH. The number of bits necessary for the tag vary according to the application. In a context where the adversary obtains something quite valuable from a successful forgery, one may wish to choose a tag length of 80 bits or more. In contexts such as authenticating a video stream, where an adversary would have to forge many frames to have a major impact on the image, an 8-bit tag may be appropriate.

FORMING R USING A BLOCK-CIPHER CALL. During our work we discovered that there are methods for authenticated-encryption which encrypt M using $\lceil |M|/n \rceil + 1$ block-cipher calls, as opposed to our $\lceil |M|/n \rceil + 2$ calls. Shai Halevi has also made this finding [17]. However, the methods we know to shave off a block-cipher call either require an unpredictable IV instead of a nonce, or they add conceptual and computational complexity to compute the initial offset R by non-cryptographic means (e.g., using a finite-field multiplication of the nonce and a key variant).

AVOIDING MOD-2^n ADDITION. Our earlier designs included a scheme based on modular 2^n addition ("addition" for the remainder of this paragraph). Basing an authenticated-encryption scheme on addition is an interesting idea due to Gligor and Donescu [15]. Compared to our GF(2^n)-based approach ("xor" for the remainder of this paragraph), an addition-based scheme is quicker to understand a specification for, and may be easier to implement. But the use of addition (where $n \geq 128$) has several disadvantages:

- The bit-asymmetry of the addition operator implies that the resulting scheme will have a bias towards big-endian architectures or little-endian architectures; there will be no way to achieve an endian-neutral scheme. The AES algorithm was constructed to be endian-neutral and we wanted OCB-AES to inherit this attribute.
- Addition is unpleasant for implementations using high-level languages, where one normally has no access to the add-with-carry instruction the machine may have.
- Addition needs more chip area than xor.
- Addition is not parallelizable. As a consequence, dedicated hardware will perform this operation more slowly than xor, and, correspondingly, modern processors can xor two n-bit quantities faster than they can add them.
- The concrete security bound appears to be worse with addition than xor (though still not bad). The degradation would seem to be $\Theta(\lg \bar{m})$, where $\bar{m}$ is the maximal message length.

We eventually came to feel that even the simplicity benefit of addition was not quite real: these schemes seem harder to understand, to prove correct, and to implement well.

LAZY MOD-p ADDITION. Let p be the largest prime less than 2^n. An earlier design [31] allowed one to produce offset $Z[i]$ from $Z[i-1]$ by "lazy mod-p addition": add L to $Z[i-1]$, mod 2^n, and then add $\delta = 2^n - p$ whenever the first addition generates a carry. Now $X[m]$ would be defined by $\mathrm{len}(M[m]) \oplus \overline{Z[m]}$, say, where $\overline{Z[m]}$ is the bitwise complement of $Z[m]$. It appears that, unlike a mod-2^n scheme, xors can still be used to combine offsets with message blocks and enciphered message blocks. This might make lazy mod-p approach more attractive than a mod-2^n approach. But in order to propagate a single scheme, avoid endian favoritism, and avoid complicating an already complex proof, we chose not to propagate lazy mod-p-addition.

DEFINITION OF THE CHECKSUM. An initially odd-looking aspect of OCB's definition is the definition of Checksum $= M[1] \oplus \cdots M[m-1] \oplus C[m]\,0^* \oplus Y[m]$. In Jutla's scheme, where one assumes that all messages are a positive multiple of the block length, the checksum is the simpler-looking $M[1] \oplus \cdots M[m-1] \oplus M[m]$. We comment that these two definitions are identical in the case that $|M[m]| = n$. What is more, the definition Checksum $= M[1] \oplus \cdots M[m-1] \oplus M[m]\,0^*$ turns out to be the wrong way to generalize the Checksum to allow for short-final-block messages; in particular, the scheme using that checksum is easily attacked.

AVOIDING PRETAG COLLISIONS. Many of our earlier schemes, including [31], allowed the adversary to force a "pretag collision." Recall that we compute the tag T by computing a "pretag" $X[m+1] = \text{Checksum} \oplus \text{SomeOffset}$, forming a value $Y[m+1] = E_K(X[m+1])$, and then forming T by doing further processing to $Y[m+1]$. For a scheme of this form, we say that an adversary can force a pretag collision if there is an N, $\bar{M}$ that can be encrypted, getting $\bar{C}\,\bar{T}$, and then a forgery attempt N, CT can be made such that, in it, the pretag $X[m+1]$ will coincide with a value $X[i]$ or $\bar{X}[i]$ at which the block cipher E was already evaluated.

We designed OCB so that an adversary can not force pretag collisions. The presence of pretag collisions substan-

tially complicates proofs, since one can not follow a line of argument that shows that tags are unpredictable because each pretag-value is almost certainly new. For schemes like IAPM, where pretag collisions can be forced, this intuition is simply wrong. Beyond this, in the presence of pretag collisions one must modify $Y[m+1]$ by an amount Δ that depends on at least the key and nonce. Say that the modification is by xor, and one wants to be able to pull off an arbitrary bit as a 1-bit authentication tag. Then every bit of Δ will have to be adversarially unpredictable. This is unfortunate, as many natural ways to make Δ fail to have this property. Suppose, for example, the first couple bits of L are forced to zero, as suggested by [31], and $\Delta = L \cdot (m+1)$. Then, for small m, the first bit of Δ will be zero. This can be exploited to give an attack on the xor-based scheme of [31] when $\tau = 1$. Similarly, for i a power of two, $\Delta = iL \bmod 2^n$ ends in a 0-bit, so had [31] taken the tag to be the last τ bits instead of the first τ bits, one would again have an attack on 1-bit tags. A scheme would be arcane, at best, if certain bits of the full tag are usable and other bits are not.

BLOCK-CIPHER CIRCUIT-DEPTH. One further efficiency measure is the circuit depth of an encryption scheme as measured in terms of block-cipher gates. For OCB encryption, this number is three: a call to form R; calls to form the ciphertext core; and a call to compute the tag. Block-cipher circuit-depth serves as a lower bound for latency in an agressively parallel environment. Reducing the block-cipher circuit-depth to one or two is possible, but the benefit does not seem worth the associated drawbacks.

5. THEOREMS

5.1 Security Definitions

We begin with the requisite definitions. These are not completely standard because OCB uses a nonce, and we wish to give the adversary every possible advantage (more than is available in real life) by allowing her to choose this nonce (though we forbid the adversary from choosing the same nonce twice).

SYNTAX. We extend the syntax of an encryption scheme as given in [3]. A (nonce-using, symmetric) encryption scheme is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and an associated number n (the nonce length). Here $\mathcal{K}$ is a finite set and $\mathcal{E}$ and $\mathcal{D}$ are deterministic algorithms. Encryption algorithm $\mathcal{E}$ takes $K \in \mathcal{K}$, $N \in \{0,1\}^n$, and $M \in \{0,1\}^*$, and returns a string $C \leftarrow \mathcal{E}_K(N, M)$. Decryption algorithm $\mathcal{D}$ takes $K \in \mathcal{K}$, $N \in \{0,1\}^n$, and $C \in \{0,1\}^*$, and returns $\mathcal{D}_K(N, M)$, which is either a string $M \in \{0,1\}^*$ or the distinguished symbol INVALID. If $C \leftarrow \mathcal{E}_K(N, M)$ then $\mathcal{D}_K(N, C) = M$.

PRIVACY. We give a particularly strong definition of privacy, one asserting indistinguishability from random strings. This notion is easily seen to imply more standard definitions [3], and by tight reductions. Consider an adversary A who has one of two types of oracles: a "real" encryption oracle or a "fake" encryption oracle. A real encryption oracle, $\mathcal{E}_K(\cdot,\cdot)$, takes as input N, M and returns $C \leftarrow \mathcal{E}_K(N, M)$. Assume that $|C| = \ell(|M|)$ depends only on $|M|$. A fake encryption oracle, $\$(\cdot,\cdot)$, takes as input N, M and returns a random string $C \xleftarrow{R} \{0,1\}^{\ell(|M|)}$. Given adversary A and encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, define $\mathbf{Adv}^{\mathrm{priv}}_{\Pi}(A) = \Pr[K \xleftarrow{R} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot)} = 1] - \Pr[K \xleftarrow{R} \mathcal{K} : A^{\$(\cdot,\cdot)} = 1]$.

An adversary A is *nonce-respecting* if it never repeats a nonce: if A asks its oracle a query (N, M) it will never subsequently ask its oracle a query (N, M'), regardless of its coins (if any) and regardless of oracle responses. All adversaries are assumed to be nonce-respecting.

AUTHENTICITY. We extend the notion of integrity of ciphertexts of [7, 8, 22]. Fix an encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and run an adversary A with an oracle $\mathcal{E}_K(\cdot,\cdot)$ for some key K. Adversary A *forges* (in this run) if A is nonce-respecting, A outputs (N, C) where $\mathcal{D}_K(N, C) \neq$ INVALID, and A made no earlier query (N, M) which resulted in a response C. Let $\mathbf{Adv}^{\mathrm{auth}}_{\Pi}(A) = \Pr[K \xleftarrow{R} \mathcal{K} : A^{\mathcal{E}_K(\cdot,\cdot)} \text{ forges }]$. We stress that the nonce used in the forgery attempt may coincide with a nonce used in one of the adversary's queries.

BLOCK CIPHERS AND PRFS. A *function family* from n-bits to n-bits is a map $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K}$ is a finite set of strings. It is a *block cipher* if each $E_K(\cdot) = E(K, \cdot)$ is a permutation. Let $\mathrm{Rand}(n)$ denote the set of all functions from $\{0,1\}^n$ to $\{0,1\}^n$ and let $\mathrm{Perm}(n)$ denote the set of all permutations from $\{0,1\}^n$ to $\{0,1\}^n$. These sets can be regarded as function families by imagining that each member is specified by a string. For $\pi \in \mathrm{Perm}(n)$, let $\pi^{-1}(Y)$ be the unique string X such that $\pi(X) = Y$. Let $\mathbf{Adv}^{\mathrm{prf}}_{E}(A) = \Pr[A^{E_K(\cdot)} = 1] - \Pr[A^{\rho(\cdot)} = 1]$, $\mathbf{Adv}^{\mathrm{prp}}_{E}(A) = \Pr[A^{E_K(\cdot)} = 1] - \Pr[A^{\pi(\cdot)} = 1]$, and $\mathbf{Adv}^{\mathrm{sprp}}_{E}(A) = \Pr[A^{E_K(\cdot), E_K^{-1}(\cdot)} = 1] - \Pr[A^{\pi(\cdot), \pi^{-1}(\cdot)} = 1]$, where the probability is over $K \xleftarrow{R} \mathcal{K}$, $\rho \xleftarrow{R} \mathrm{Rand}(n)$, and $\pi \xleftarrow{R} \mathrm{Perm}(n)$.

5.2 Theorem Statements

We give information-theoretic bounds on the authenticity and the privacy of OCB. Proofs are in the full paper [33].

THEOREM 1. *Fix OCB parameters n and τ. Let A be an adversary that asks q queries and then makes its forgery attempt. Suppose the q queries have aggregate length of σ blocks, and the adversary's forgery attempt has at most c blocks. Let $\bar{\sigma} = \sigma + 2q + 5c + 11$. Then*

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) \;\leq\; \frac{1.5\,\bar{\sigma}^2}{2^n} + \frac{1}{2^\tau}$$

The aggregate length of queries $M_1, \ldots, M_q$ means the number $\sigma = \sum_{r=1}^{q} \|M_r\|_n$.

It is standard to pass to a complexity-theoretic analog of Theorem 1, but in doing this one will need access to an E^{-1} oracle in order to verify a forgery attempt, which translates into needing the strong PRP assumption. One gets the following. Fix OCB parameters n and τ, and a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Let A be an adversary that asks q queries and then makes its forgery attempt. Suppose the q queries have aggregate length of σ blocks, and the adversary's forgery attempt has at most c blocks. Let $\bar{\sigma} = \sigma + 2q + 5c + 11$. Let $\delta = \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[E,\tau]}(A) - 1.5\,\bar{\sigma}^2/2^n - 1/2^\tau$. Then there is an adversary B for attacking block cipher E that achieves advantage $\mathbf{Adv}^{\mathrm{sprp}}_{E}(B) \geq \delta$. Adversary B asks at most $q' = \sigma + 2q + c + 3$ oracle queries and has a running time which is equal to A's running time plus the time to compute E or E^{-1} at q' points plus additional time which is $\alpha n \bar{\sigma}$, where the constant α depends only on details of the model of computation.

The privacy of OCB is given by the following result.

THEOREM 2. *Fix OCB parameters n and τ. Let A be an adversary that asks q queries, these having aggregate length of σ blocks. Let $\bar{\sigma} = \sigma + 2q + 3$. Then*

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathrm{OCB}[\mathrm{Perm}(n),\tau]}(A) \leq \frac{1.5\,\bar{\sigma}^2}{2^n}$$

It is standard to pass to a complexity-theoretic analog of Theorem 2. One gets the following. Fix OCB parameters n and τ, and a block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. Let A be an adversary that asks q queries, these having aggregate length of σ blocks. Let $\bar{\sigma} = \sigma + 2q + 3$. Let $\delta = \mathbf{Adv}^{\mathrm{auth}}_{\mathrm{OCB}[E,\tau]}(A) - 1.5\,\bar{\sigma}^2/2^n$. Then there is an adversary B for attacking block cipher E that achieves advantage $\mathbf{Adv}^{\mathrm{prp}}_E(B) \geq \delta$. Adversary B asks at most $q' = \sigma + 2q + 1$ oracle queries and has a running time which is equal to A's running time plus the time to compute E at q' points plus additional time which is $\alpha n\bar{\sigma}$, where the constant α depends only on details of the model of computation.

5.3 Proofs

To prove Theorem 1 (Theorem 2 is comparatively easy) we give a *structure lemma* that relates the authenticity of OCB to three functions: the M-collision probability, the MM-collision probability, and the CM-collision probability. Very informally, these measure the probability of trouble when the adversary asks a single query, some pair of queries, and when adversary tries to forge some ciphertext following a single earlier query. Due to space limitations, all of these definitions, lemmas and proofs are removed from this proceedings paper. They can be found in the full paper [33].

6. PERFORMANCE

ABSTRACT ACCOUNTING. OCB uses $\lceil |M|/n \rceil + 2$ block-cipher calls to encrypt a nonempty message M. (The empty string takes three block-cipher calls.) We compare this with CBC encryption and CBC encryption plus a CBC MAC:

- "Basic" CBC encryption, where one assumes a random IV and a message which is a multiple of the block length, uses two fewer block-cipher calls—a total of $|M|/n$.
- A more fair comparison sets IV $= E_K(N)$ for CBC encryption (so both schemes use a not-necessarily-random nonce), and uses obligatory 10^* padding (so both schemes can handle arbitrary strings). This would bring the total for CBC to $\lceil (|M|+1)/n \rceil + 1$ block-cipher calls, coinciding with OCB in the case that $|M|$ is a multiple of the block length, and using one fewer block-cipher call otherwise.
- If one combines the basic CBC encryption with a MAC, say MACing the ciphertext, then the CBC-encryption will use a number of block-cipher calls as just discussed, while the CBC MAC will use between $\lceil |M|/n \rceil + 1$ and $\lceil (|M|+1)/n \rceil + 3$ block-cipher calls, depending on padding conventions and the optional processing done to the final block in order to ensure security across messages of varying lengths. So the total will be as few as $2\lceil |M|/n \rceil + 1$ or as many as $2\lceil (|M|+1)/n \rceil + 4$ block-cipher calls. Thus OCB saves between $\lceil |M|/n \rceil - 1$ and $\lceil |M|/n \rceil + 3$ block-cipher calls compared to separate CBC encryption and CBC MAC computation

As with any mode, there is overhead beyond the block-cipher calls. Per block, this overhead is about four n-bit xor operations, plus associated logic. The work for this associated logic will vary according to whether or not one precomputed $L(i)$-values and many additional details.

Algorithm	64 B	256 B	1 KB	4 KB
OCB encrypt	24.7	18.5	**16.9**	16.7
ECB encrypt	15.1	15.0	**14.9**	14.9
CBC encrypt	15.9	15.9	**15.9**	15.9
CBC mac	19.2	16.3	**15.5**	15.3

Figure 3: Performance results from Lipmaa [25], in cycles per byte on a Pentium III. The block cipher is AES128. Code is written in assembly.

Though some of the needed $L(i)$-values are likely to be precomputed, computing all of them "on the fly" is not inefficient. Starting with 0^n we form successive offsets by xoring the previous offset with L, $2 \cdot L$, L, $4 \cdot L$, L, $2 \cdot L$, L, $8 \cdot L$, and so forth. So half the time we use L itself; a quarter of the time we use $2 \cdot L$; one eighth of the time we use $4 \cdot L$; and so forth. Thus the expected number of times to multiply by x in order to compute an offset is at most $\sum_{i=1}^{\infty} i/2^{i+1} = 1$. Each $a \cdot \mathrm{x}$ instruction requires an n-bit shift and a conditional 32-bit xor. Said differently, for any $m > 0$, the total number of $a \cdot \mathrm{x}$ operations needed to compute $\gamma_1 \cdot L$, $\gamma_2 \cdot L$, ..., $\gamma_m \cdot L$ is $\sum_{i=1}^{m} \mathrm{ntz}(i)$, which is less than m. The above assumes that one does not retain or precompute any $L(i)$ value beyond $L = L(0)$. Suppose that one precomputes $L(-1)$, $L(0)$, $L(1)$, $L(2)$, $L(3)$. Computing and saving the four values beyond $L = L(0)$ is cheaper than computing L itself, which required an application of E_K. But now the desired multiple of L will have be available at least $1/2 + 1/4 + 1/8 + 1/16 \approx 94\%$ of the time. When it has not been precomputed it must be calculated, starting from $L(3)$, so the amortized number of multiplications by x has been reduced to $\sum_{i=1}^{\infty} = i/2^{i+4} = 0.125$.

EXPERIMENTAL RESULTS. In Table 3 we report, with permission, some experimental results by Helger Lipmaa [25]. On a Pentium III, in optimized assembly, Lipmaa implemented OCB encryption, ECB encryption, CBC encryption, and the CBC MAC. The last three modes were implemented in their "raw" forms, where one does no padding and assumes that the message acted on is a positive multiple of the block length. For CBC encryption, the IV is fixed. The underlying block cipher is AES128.

Focusing on messages of 1 KByte, OCB incurs about 6.4% overhead compared to CBC encryption, and the algorithm takes about 54% of the time of a CBC encryption + CBC MAC. Lipmaa points out that overhead is so low that, in his experiments, an assembly AES128 with a C-code CBC-wrapper is slightly slower than the same AES128 with an assembly OCB-wrapper. Lipmaa's (size-unoptimized) code is 7.2 KBytes, which includes unrolling AES128 (2.2 KBytes) three times.

Some aspects of the experiments above are unfavorable to OCB, making the performance estimates conservative. In particular, the "raw" CBC MAC needs to be modified to correctly handle length-variability, and doing so is normally done in a way that results in additional block-cipher calls. And when combined with CBC encryption, the CBC MAC should be taken over the full ciphertext, including the nonce, which would add an extra block-cipher call. Finally, an extra

block-cipher call would normally be performed by CBC to correctly compute the IV from a nonce.

The results above are for a serial execution environment. In settings with plenty of registers and multiple instruction pipes, OCB, properly implemented, will be faster than CBC.

Acknowledgments

At CRYPTO '00, Virgil Gligor described [12] to Rogaway, Charanjit Jutla gave a rump-session talk on [18], and Elaine Barker announced a first modes-of-operation workshop organized by NIST. These events inspired [31], which evolved into the current work. After the first workshop NIST made a second call for proposals, and OCB took its final form in response to this call [32]. We appreciate NIST's effort to solicit and evaluate modern modes of operation. Elaine Barker, Morris Dworkin, and Jim Foti are among those involved.

We received useful feedback from Michael Amling, Paulo Barreto, Johan Håstad, Hugo Krawczyk, Helger Lipmaa David McGrew, Silvio Micali, Ilya Mironov, Alberto Pascual, Bart Preneel, Tom Shrimpton, and David Wagner. Special thanks to Michael and Ilya for their careful proofreading, and Helger for doing a state-of-the-art assembly implementation, along with providing associated timing data. Thanks to the anonymous CCS referees for their comments.

This work was carried out while Rogaway was on leave of absence from UC Davis, visiting the Department of Computer Science, Faculty of Science, Chiang Mai University. No external funding has been used for this research, but the submitter gratefully acknowledges a recent gift by Cisco Systems. Many thanks to Cisco for their support.

7. REFERENCES

[1] J. An and M. Bellare. Does encryption with redundancy provide authenticity? *Advances in Cryptology – EUROCRYPT 2001.* Lecture Notes in Computer Science, vol. 2045, B. Pfitzmann, ed., Springer-Verlag, 2001. www-cse.ucsd.edu/users/mihir

[2] K. Aoki and H. Lipmaa. Fast implementations of AES candidates. Third AES Candidate Conference, New York City, USA, Apr 2000, pp. 106–120. www.tml.hut.fi/~helger

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. *Proceedings of 38th Annual Symposium on Foundations of Computer Science* (FOCS 97), IEEE, 1997. www.cs.ucdavis.edu/~rogaway

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – CRYPTO '98.* Lecture Notes in Computer Science, vol. 1462, H. Krawczyk, ed., Springer-Verlag. www.cs.ucdavis.edu/~rogaway

[5] M. BELLARE, R. GUÉRIN, AND P. ROGAWAY. "XOR MACs: New methods for message authentication using finite pseudorandom functions." *Advances in Cryptology – CRYPTO '95.* Lecture Notes in Computer Science, vol. 963, Springer-Verlag, D. Coppersmith, ed., pp. 15–28, 1995. www.cs.ucdavis.edu/~rogaway

[6] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, vol. 61, no. 3, Dec 2000.

[7] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology – ASIACRYPT '00.* Lecture Notes in Computer Science, vol. 1976, T. Okamoto., ed., Springer-Verlag, 2000. www-cse.ucsd.edu/users/mihir

[8] M. Bellare and P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient encryption. *Advances in Cryptology – ASIACRYPT '00.* Lecture Notes in Computer Science, vol. 1976, T. Okamoto., ed., Springer-Verlag, 2000. www.cs.ucdavis.edu/~rogaway

[9] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS #1. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science, vol. 1462, pp. 1–12, 1998. www.bell-labs.com/user/bleichen

[10] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. on Comp.*, vol. 30, no. 2, pp. 391–437, 2000.

[11] V. Gligor and P. Donescu. Integrity-aware PCBC encryption schemes. *Security Protocols, 7th International Workshop, Cambridge, UK, April 1999 Proceedings.* Lecture notes in Computer Science, vol. 1796, Springer-Verlag, pp. 153–171, 2000.

[12] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Manuscript, Aug 18, 2000. Formerly available from www.eng.umd.edu/~gligor.

[13] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Contribution to NIST, Oct 27, 2000. csrc.nist.gov/encryption/aes/modes

[14] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. Contribution to NIST. Mar 30, 2001, rev. Apr 20, 2001. csrc.nist.gov/encryption/modes/proposedmodes

[15] V. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. *Fast Software Encryption*, Lecture Notes in Computer Science, Springer-Verlag, Apr 2001.

[16] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, vol. 28, Apr 1984, pp. 270–299.

[17] S. Halevi. An observation regarding Jutla's modes of operation. Cryptology ePrint archive, reference number 2001/015, submitted Feb 22, 2001, revised Apr 2, 2001. eprint.iacr.org

[18] C. Jutla. Encryption modes with almost free message integrity. Cryptology ePrint archive, report 2000/039, Aug 1, 2000. eprint.iacr.org

[19] C. Jutla. Encryption modes with almost free message integrity. Contribution to NIST. Undated manuscript, appearing Oct 2000 at csrc.nist.gov/encryption/modes/workshop1

[20] C. Jutla. Encryption modes with almost free message integrity. Contribution to NIST. Posted May 24, 2001 at csrc.nist.gov/encryption/modes/proposedmodes

[21] C. Jutla. Encryption modes with almost free message integrity. *Advances in Cryptology – EUROCRYPT 2001*. Lecture Notes in Computer Science, vol. 2045, B. Pfitzmann, ed., Springer-Verlag, 2001.

[22] J. Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. *Fast Software Encryption '00*. Lecture Notes in Computer Science, B. Schneier, ed., 2000.

[23] J. Katz and M. Yung. Complete characterization of security notions for probabilistic private-key encryption. STOC 2000, pp. 245–254, 2000.

[24] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). Advances in Cryptology — CRYPTO '01. Springer-Verlag, 2001. Earlier version as ePrint report 2001/045, Jun 6, 2001. eprint.iacr.org/20001/045

[25] H. Lipmaa. Personal communications, Jul 2001. Further information at www.tcs.hut.fi/~helger

[26] M. Luby and C. Rackoff. How to construct pseudororandom permutations from pseudorandom functions. *SIAM J. Computation*, vol. 17, no. 2, Apr 1988.

[27] M. Matyas and S. Matyas. *Cryptography: A new dimension in computer data security*. John Wiley & Sons, New York, 1982.

[28] RSA Laboratories. PKCS #1: RSA encryption standard, Version 1.5, Nov 1993; and PKCS #1: RSA cryptography specifications, Version 2.0, Sep 1998, B. Kaliski and J. Staddon. www.rsasecurity.com/rsalabs/pkcs/pkcs-1

[29] J. Steiner, C. Neuman, and J. Schiller. Kerberos: an authentication service for open network systems. *Proceedings of the Winter 1988 Usenix Conference*, pp. 191–201, 1988.

[30] B. Preneel. Cryptographic primitives for information authentication — State of the art. *State of the Art in Applied Cryptography*, COSIC '97, LNCS 1528, B. Preneel and V. Rijmen, eds., Springer-Verlag, pp. 49–104, 1998.

[31] P. Rogaway. OCB mode: Parallelizable authenticated encryption. Contribution to NIST, Oct 16, 2000. (Preliminary version of the OCB algorithm.) csrc.nist.gov/encryption/modes/workshop1

[32] P. Rogaway (submitter) and M. Bellare, J. Black, and T. Krovetz (auxiliary submitters). OCB mode. Contribution to NIST. Cryptology ePrint archive, report 2001/26, Apr 1, 2001, revised Apr 18, 2001. ePrint.iacr.org and csrc.nist.gov/encryption/modes/proposedmodes.

[33] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. Full version of this paper. Aug 2001. www.cs.ucdavis.edu/~rogaway

[34] US National Institute of Standards. Specification for the Advanced Encryption Standard (AES). Draft Federal Information Processing Standards, Feb 28, 2001. Based on: J. Daemen and V. Rijmen, AES Proposal: Rijndael. Sep 3, 1999. www.nist.gov/aes

APPENDIX

A. BRIEF HISTORY

JUTLA, GLIGOR-DONESCU, ROGAWAY. An April 1999 paper by Gligor and Donescu gives an authenticated-encryption scheme called PCBC [11]. The mode is wrong, as pointed out by Jutla [18], but it may have contributed to the subsequent development of correct modes. Jutla's paper [18] gives the first apparently correct schemes, IACBC and IAPM. Shortly after that paper appeared, Gligor and Donescu described a different scheme, XCBC [12], which is similar to IACBC. The most conspicuous difference between XCBC and IACBC is the former's use of mod-2^n addition where the latter uses xor or mod-p addition, for p a prime just less than 2^n.

A first call by NIST for modes of operation brought contributions [13, 19] based on [12, 18], and a contribution by Rogaway [31] that built on [18]. In [19], Jutla employs a Gray-code ordering for combining basis offsets, a refinement independently introduced, along with further tricks, in [31].

A second call by NIST gave rise to [14, 20, 32], which were revisions to [13, 19, 31], respectively. In [20], Jutla emphasized IAPM over IACBC, and he adopted lazy mod-p addition, first described in [31]. In [14], Gligor and Donescu describe four authenticated-encryption modes, one of which, XECBS-XOR, is parallelizable. The modes adopt some features introduced in [31] to deal with messages of arbitrary length and to use a single block-cipher key. In [32], Rogaway et al. settled on one mechanism to make offsets (three are described in [31]) and made further refinements to [31].

Briefly comparing OCB and IAPM, the latter uses two separate keys and is defined only for messages which are a multiple of the block length. Once a padding regime is included, say obligatory 10^* padding, ciphertexts will be longer than OCB's by 1 to n bits. IAPM supports offset-production using either lazy mod-p addition or an xor-based scheme. The latter is not competitive with OCB in terms of session-setup costs.

The initial version of Jutla's work [18] claimed a proof, and included ideas towards one. A subsequent writeup by Halevi [17] was more rigorous.

PATENTS. The history above ignores associated patent applications. Jutla/IBM, Gligor/VDG, and Rogaway have all indicated that there were such filings. All parties have provided statements to NIST promising reasonable and nondiscriminatory licensing.

DEFINITIONS. Though the authenticated-encryption goal is folklore, provable-security treatments of it are recent. The first definition for authenticated encryption is due to Bellare and Rogaway [8] and, independently, Katz and Yung [22]. Bellare and Namprempre were the first to seriously investigate the properties of authenticated-encryption and the generic-composition paradigm [7].